

Requested Patent: JP6202996A

Title:

EXPANSION FUNCTION OF SUBSCRIBER-DISTRIBUTED TWO-PHASE COMMIT
PROTOCOL ;

Abstracted Patent: JP6202996 ;

Publication Date: 1994-07-22 ;

Inventor(s):

BROCKMEYER ROGER L;; DIEVENDORFF RICHARD;; HOUSE DANIEL E;;
JENNER EARLE H;; LABELLE MARGARET K;; MALL MICHAEL G;; SILEN STUART
L ;

Applicant(s): INTERNATL BUSINESS MACH CORP It;IBMgt; ;

Application Number: JP19930202200 19930816 ;

Priority Number(s): ;

IPC Classification: G06F15/16; G06F13/00 ;

Equivalents: JP2675968B2 ;

ABSTRACT:

PURPOSE: To provide an expansion function of two-phase commit protocol which attains the subscription of distributed subscribers between physically separated agents without relying upon the communication mechanism used by data processing systems.

CONSTITUTION: A special processing state called first-phase (EPOP) which can distribute coordinator functions by using an arbitrary communication mechanism is added to a two-phase commit protocol. A subscriber sends a two-phase commit protocol sequence to a distributed system in a special stage in which the subscriber can acquire a control right. The communication mechanism is used so that the mechanism can become part of a distributed coordinator. The coordinator itself does not require the knowledge about the other systems. The special processing system is validated by operating system service called first-phase terminating process enable (EEPOEP). Because of the EEPOEP, the expansion function of the two-phase commit protocol can be used on an executing-side system.

SVL920030072057

特開平6-202996

(43)公開日 平成6年(1994)7月22日

(51)Int.Cl. ⁵	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 15/16	3 7 0 M	7429-5L		
13/00	3 5 5	7368-5B		

審査請求 有 請求項の数12 (全 29 頁)

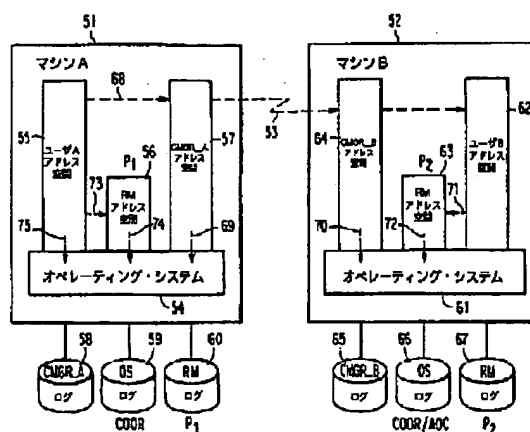
(21)出願番号	特願平5-202200	(71)出願人	390009531 インターナショナル・ビジネス・マシーンズ・コーポレーション INTERNATIONAL BUSINESS MACHINES CORPORATION アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)
(22)出願日	平成5年(1993)8月16日	(72)発明者	ロジャー・ライル・ブロックマイヤー アメリカ合衆国95139、カリフォルニア州 サン・ホセ、スカウヘイガン・コート 148
(31)優先権主張番号	9 3 2 8 3 5	(74)代理人	弁理士 合田 潔 (外3名)
(32)優先日	1992年8月20日		最終頁に続く
(33)優先権主張国	米国 (US)		

(54)【発明の名称】 加入者分散2相コミット・プロトコルの拡張機能

(57)【要約】 (修正有)

【目的】 データ処理システムで用いられている通信メカニズムに依存せずに、物理的に離れたエージェント間で分散加入を可能にする2相コミット・プロトコルの拡張機能を提供する。

【構成】 任意の通信メカニズムを使用してコーディネータ機能を分散させることができる第1相終了処理 (EPOP) と呼ばれる特別処理ステージが2相コミット・プロトコルに追加される。加入者が制御権を得られる特別ステージで、2相コミット・プロトコル・シーケンスを分散システムに送る。通信メカニズムは、それが分散コーディネータの一部になるように用いられる。コーディネータそれ自体は他のシステムの知識を要しない。特別処理ステージは、第1相終了処理イネーブル (EEOEP) と呼ばれるオペレーティング・システム・サービスによって有効になる。EEOEPにより、2相コミット・プロトコルの拡張機能が実行側システム上で使用できる。



【特許請求の範囲】

【請求項1】分散データ処理システムにおける更新調整方法であって、

上記分散データ処理システム内の第1システム上で第1相終了処理を有効にし、上記第1システム上の第1コーディネータ機能が、上記分散データ処理システム内の第2システム上の第2コーディネータ機能のエージェントになるようにするステップと、

上記第1システム上のすべての加入者からの投票がすべて受信された時に、第1システムの投票の第1相終了レコードを上記第1コーディネータがログに書込むステップと、

上記書込みに応答して制御を上記第1相終了処理に引渡すステップと、

を含む更新調整方法。

【請求項2】準備のための上記投票が、加入者が作業単位の変更をCOMMITする準備が出来たことを示すYES、加入者が作業単位の変更をバックアウトしようとすることを示すNO、または加入者が作業単位に継続して関わることを望み、作業単位の最終決定には影響を与えようとしな

【請求項3】少なくとも第1オペレーティング・システムを実行する第1マシンと、第2オペレーティング・システムを実行する第2マシンとを含む分散データ処理システムにおける更新調整方法であって、

上記第1オペレーティング・システムが、上記第2マシン上の第2通信マネージャと通信する第1通信マネージャを含む上記第1マシン上の加入者に準備信号を送り、上記第1通信マネージャが上記準備信号を上記第2通信マネージャに送るステップと、

上記第2通信マネージャが、上記第1通信マネージャからの準備信号に応答して、上記第2オペレーティング・システムに対して第1相終了イネーブル・コールを生成し、上記第2オペレーティング・システムが上記第1オペレーティング・システムのコーディネータ機能のエージェントになるようにするステップと、

上記第2オペレーティング・システムが、上記第2通信マネージャを含む上記第2マシン上の加入者に準備信号を送るステップと、

上記第2マシン上の上記加入者から投票を受信するステップと、

第1相終了処理により、上記第1通信マネージャに上記投票を送る上記第2通信マネージャに制御を渡すステップと、

を含む更新調整方法。

【請求項4】準備に応答する上記投票が、加入者が作業単位の変更をCOMMITする準備が出来たことを示すYES、加入者が作業単位の変更をバックアウトしよう

とすることを示すNO、または加入者が作業単位に継続して関わることを望み、作業単位の最終決定には影響を与えようとしな

【請求項5】上記第2マシン上の上記加入者からの上記投票の少なくとも一部がYESであり、上記第2オペレーティング・システムが、上記第2マシン上の上記加入者による投票の第1相終了レコードを第2マシン・ログに書込むステップを含み、上記第2通信マネージャに制御を引渡す上記ステップが、上記第2マシンログへの書込み後に第1相終了処理によって実行される、請求項4記載の分散データ処理システムにおける更新調整方法。

【請求項6】上記第2マシン上の加入者による上記投票のすべてがFORGETまたはABSTAINであり、上記第1オペレーティング・システムが、上記第1通信マネージャによって上記第2通信マネージャから受信された投票を含むすべての投票を上記第1マシン上の加入者から受信して、アトミック・インスタント・レコードを第1マシン・ログに書込むステップと、

上記第1オペレーティング・システムが、上記投票の結果によって決定された実行信号を上記第1マシン上の加入者に送り、上記第2マシン上の上記加入者による上記FORGETまたはABSTAINの投票により上記第2通信マネージャには実行信号が送られないステップと、を含む請求項4記載の分散データ処理システムにおける更新調整方法。

【請求項7】上記第1マシン上の加入者による準備のための投票がすべてFORGETまたはABSTAINであり、

上記第2オペレーティング・システムが、上記第2マシン上の上記加入者による投票の第1相終了レコードを第2マシン・ログに書込むステップと、

上記第2マシン・ログへの書込みの後、上記第1通信マネージャに上記投票を送る上記第2通信マネージャに制御を渡すステップと、

上記第1オペレーティング・システムが、上記第1通信マネージャによって上記第2通信マネージャから受信された投票を受信して、アトミック・インスタント・レコードを第1マシン・ログに書込むステップと、

上記第1オペレーティング・システムが、FORGETまたはABSTAINを投票した、上記第1マシン上の加入者には実行信号を送らず、上記第1通信マネージャに送り、上記第1通信マネージャが上記第2通信マネージャに上記実行信号を送るステップと、

上記第2オペレーティング・システムが、上記第2通信マネージャによって受信された上記実行信号に応答して上記第2マシン・ログに実行レコードを書込むステップと、

上記第2通信マネージャが、実行が完了したとの通知を上

記第1通信マネジャに送るステップと、
を含む更新調整方法。

【請求項8】上記第1オペレーティング・システムが、
上記第1通信マネジャによって上記第2通信マネジャから
受信された投票を含む上記第1マシン上の加入者からの
投票をすべて受信して、アトミック・インスタント・
レコードを第1マシン・ログに書込むステップと、
上記第1オペレーティング・システムが、上記投票の結果
によって決定された実行信号を上記第1マシン上の加入
者に送り、上記第1通信マネジャが上記実行信号を上
記第2通信マネジャに送るステップと、
上記第2オペレーティング・システムが、上記第2通信
マネジャによって受信された上記実行信号にตอบสนองして上
記第2マシン・ログに実行レコードを書込むステップと、
実行が完了したことの通知を上記第2通信マネジャが上
記第1通信マネジャに送るステップと、
を含む請求項4記載の分散データ処理システムにおける
更新調整方法。

【請求項9】上記第2オペレーティング・システムが、
上記第2マシン上の加入者から実行が完了したとの確認
応答を受信して、上記第2マシン上の加入者に上記実行
信号を送り、上記第2マシン・ログを更新し、上記第1
オペレーティング・システムが、上記第1通信マネジャ
を含む上記第1マシン上の加入者から実行が完了したと
の確認応答を受信して、上記第1マシン・ログを更新す
るステップを含む請求項8記載の分散データ処理システ
ムにおける更新調整方法。

【請求項10】拡張2相コミット・プロトコルを実現す
る分散データ処理システムであって、

第1オペレーティング・システムを実行し、第1通信マ
ネジャを含む複数の第1加入者をサポートする少なくとも
第1マシンと、

第2オペレーティング・システムを実行し、第2通信マ
ネジャを含む複数の第2加入者をサポートする少なくとも
第2マシンと、

上記第1及び第2の通信マネジャを相互接続する通信手
段とを含み、

上記第1オペレーティング・システムが、上記第1マシ
ン上の上記複数の第1加入者に準備信号を送り、上記第
1通信マネジャが上記準備信号を上記第2通信マネジャ
に送り、上記第2通信マネジャが、上記第1通信マネジャ
からの準備信号にตอบสนองして、上記第2オペレーティン
グ・システムに対して第1相終了イネーブル・コールを
生成し、上記第2オペレーティング・システムが上記第
1オペレーティング・システムのコーディネータ機能の
エージェントになるようにし、上記第2オペレーティン
グ・システムが、上記第2マシン上の加入者に準備信号
を送り、上記第2オペレーティング・システムが、上記
第2マシン上の上記加入者による投票の第1相終了レコ

ードを第2マシン・ログに書込み、上記第2マシン・ロ
グへの書込みの後、第1相終了処理により制御を上記第
2通信マネジャに渡し、上記第2通信マネジャが上記投
票を上記第1通信マネジャに送る、

分散データ処理システム。

【請求項11】上記第1及び第2のマシンが、仮想マシ
ン・オペレーティング・システム上で1つのコンピュー
タを実行する仮想マシンであり、上記第1及び第2のオ
ペレーティング・システムが、上記仮想マシン・オペレ
ーティング・システムのゲスト・オペレーティング・シ
ステムである、請求項10記載の分散データ処理システ
ム。

【請求項12】上記第1及び第2のマシンが物理的に離
れたコンピュータ上に実現され、上記通信装置が上記コ
ンピュータを相互接続する通信リンクを含む、請求項1
0記載の分散データ処理システム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、一般的にはデータ処理
システムに関し、特に加入者が遠隔加入者と相談するた
めに決定プロセスを拡張できるように、或いは他のロー
カル操作が実行できるようにする2相コミット・プロト
コルの拡張機能に関する。本発明は、通信マネジャを用
いて、分散したコミット・スコープを実現できるシステ
ムの開発に用いることができる。ユーザは分散可能であ
り、加入者も同様に分散可能である。

【0002】

【従来の技術】標準的な2相コミット・プロトコルは、
集中管理プロセス（すなわちコーディネータ）がサテラ
イト・プロセス（すなわち加入者）によるデータ変更の
結果の“硬化”を同期化することができる周知の方法であ
る。2相とは“投票”相（すなわちPREPARE信号を
加入者に送ること）と“実行”相（すなわちCOMMIT
またはBACKOUT信号を加入者に送ること）であ
る。

【0003】加入者は、復元可能なデータを変更した時
はコーディネータに通知しなければならない。データが
コミットされる準備が出来ると（すなわちユーザがCO
MMITまたはBACKOUTを要求していると）、コ
ーディネータは、2相コミット・プロトコルを使用し
て、すべての加入者が前へ進む（すなわちデータをコミ
ットする）かまたは後へ進む（すなわちデータの変更を
バックアウトする）ようにしなければならない。コー
ディネータは、全加入者の実行が同じになるようにしな
ければならない。ある加入者がコミットしてある加入者
がバックアウトすることは全く許されない。現在の2相コ
ミット・プロトコルでは、複数の加入者間でデータの保
全性が維持されるのは、データに変更を加える前に全加
入者の同意をとりつけることによる。

【0004】2相コミット・プロトコルの詳細について

は下記の文献を参照されたい。

Cruz, R. , Data Recovery in IBM Database 2, IBM Systems Journal, vol. 23, no. 2, 1984.

Date, C. J. , An Introduction to Database Systems, vol. II, Addison-Wesley (1983) .

Gray, J. N. , Notes on Data Base Operating Systems, IBM Research Report RJ2188, Feb. 1978.

Haerder, T. , and A. Reuter, Principles of Transaction-Oriented Database Recovery, ACM Computing Surveys, vol. 15, no. 4, December 1983.

Lindsay, B. , and C. Mohan, Efficient Commit Protocols for the Tree of Processes Model of Distributed Transactions, IBM Research Report RJ3881, June 2, 1983.

【0005】2相コミット・プロトコルは分散データベースに応用されているが、その成果は限られる。従来技術では、分散データベースを扱うために集中管理とリニアの両方式の2相コミット・プロトコルが検討されている。一例として、C. J. Date, B. J. LindsayらによるNotes on Distributed Databases, IBM Research Report RJ2571, July 1979を参照されたい。この文献では323ページ以降で分散データベースに用いるコミット・プロトコルが解説されている。

【0006】システム上の遠隔加入者がさまざまな通信媒体によって接続された分散データ処理システムに、2相コミット・プロトコルを応用する際には1つ問題がある。分散環境の場合、コーディネータの実行は、各種の通信媒体によって接続できるシステム全体に行き渡るようにしなければならない。現在、このような環境の2相コミット・プロトコルは実現が容易ではない。既存の分散コミット・スコープ・システムは通常、1つの通信マネージャに依存してコーディネータとして機能するか、または分散データベース・マネージャに限定される。また、分散処理システムの各サイトのコーディネータは、システム内の全サイトのIDを認識する必要があり、コーディネータが更に複雑になる。

【0007】分散システムの2相コミット・プロトコルに関しては、より柔軟なアプローチも望まれる。リソース・マネージャは、コーディネータからのPREPARE信号にある方法でしか応答しない。これらはCOMMITへの投票を示すYESか、BACKOUTへの投票を示すNO、または、基本的な2相コミット・プロトコルに対して最適化された場合は、ファイルがREAD ONLYであることを示すFORGETである。しかしコーディネータが、通信リソース・マネージャ (APPC/MVSなど) に依存しない時は、これらの応答は適切ではないことがある。YES/COMMIT応答により、コーディネータはコミットするリソースが他にない場合でも、作業単位についてログ・レコードの書込みを余儀なくされる。これにより性能は落ち、システム・スルー

ットが下がり、システム再開時間が長くなる。NO/COMMIT応答では、作業単位がバックアウトされる。FORGET/READ ONLY応答では、通信リソース・マネージャが、その必要があっても作業単位に関係しなくなる。

【0008】

【発明が解決しようとする課題】本発明の目的は、通信マネージャを使用して、同種または異種オペレーティング・システム間に分散コミット・スコープを提供することである。

【0009】本発明の目的は、データ処理システムで用いられている通信メカニズムに依存せずに、物理的に離れたエージェンツ間で分散加入を可能にする2相コミット・プロトコルの拡張機能を提供することである。

【0010】本発明の目的は、物理的に離れたエージェンツが存在する時に、2相コミット・プロトコルの性能を最適化することである。

【0011】

【課題を解決するための手段】本発明に従って、2相コミット・プロトコルに特別処理ステージが追加される。この特別ステージは第1相終了処理 (EPOP) と呼ばれ、任意の通信メカニズムを使用してシステム全体にコーディネータ機能を分散させることができる。第1相終了処理は加入者が制御権を得られる特別ステージである。この特別ステージでは、加入者が2相コミット・プロトコルのシーケンスを分散システムに送る。通信メカニズムは、これが分散コーディネータの一部になるように用いられる。コーディネータ自体は他のシステムの知識を要しない。

【0012】特別処理ステージは、第1相終了処理イネーブル (EEPOEP) と呼ばれるオペレーティング・システムのサービスによって有効になる。特別ステージは、EEPOEPにより実行側システム上で使用できる。これにより、分散データベースをサポートだけでなく、分散したユーザと分散した汎用リソース・マネージャ (障害に耐える保護リソースを所有するプロセスであるリソース・マネージャなど) もサポートされる。

【0013】本発明の態様に従って、コーディネータからのPREPARE信号に応答して新しい応答がリソース・マネージャによって使用できる。この新しい応答はABSTAINと呼ばれる。リソース・マネージャからコーディネータへのこの応答は、リソース・マネージャが作業単位の2相コミット・プロセスに関係し続けようとするが、作業単位の最終決定 (すなわちCOMMITまたはBACKOUT) には影響を与えようとしなかったことを示す。ABSTAIN応答を使用することで、本発明に従った分散された2相コミット・プロトコルの性能が更に最適化される。

【0014】

【実施例】各図を参照する。図1乃至図4は、仮想端末

アクセス方式 (VTAM) 12とジョブ入力システム (JES) 13がインストールされたメインフレーム・コンピュータ11から成るトランザクション処理システムを示す。図に示したVTAM12はアドレス空間であり、そのアドレス空間内に、直接アクセス記憶装置 (DASD) 15と通信するアクセス方式ルーチンがある。DASDにはデータベースとプログラム・ライブラリが格納される。VTAM12はローカル端末16とリモート端末17を伴う。システムのオンライン・ユーザは、複数のローカル端末16とリモート端末17を介してデータにアクセスする。ジョブ入力システム (JES) 13は、ローカル・ジョブ入力/出力 (I/O) 装置18とリモート・ジョブI/O装置19の両方と通信する。

【0015】VTAM12とJES13はいずれも、コンピュータ11にインストールされる基本オペレーティング・システムと通信する。オペレーティング・システムは、システム・リソースを中央コンソール22や、機能コンソール23、及びJES13と共有するマスタ・スケジューラ21を含む。JES13は、オペレーティング・システムの一部であるジョブ・スケジューラ24に入力されるジョブ・キューを生成する。このキューはVTAM12に入力される。

【0016】オペレーティング・システムは、たとえばIBMの多重仮想記憶 (MVS) オペレーティング・システムである。MVSオペレーティング・システムは、タスク・スーパーバイザ25、プログラム・マネジャ26、タイマ・スーパーバイザ27、仮想記憶マネジャ28、実記憶マネジャ29、補助 (またはページ) 記憶マネジャ30、復元終了マネジャ31、入力/出力システム (IOS) 機能32などを含む。これらはすべて、各種の割込みハンドラ34からの入力を受信するシステム状態機能33と通信し、出力をシステム・リソース・マネジャ35に提供する。また、タスク・スーパーバイザ25は、ジョブ・スケジューラ24及びディスクパッチャ36の両方と通信する。ディスクパッチャ36は、中央処理装置 (CPU) サービスのためにキューイングされたリクエストを持つ。

【0017】以上は、MVSオペレーティング・システムの概略である。MVSオペレーティング・システムの詳細についてはH. Lorin及びH. M. DeitelによるOperating Systems, Addison-Wesley (1981)、並びにH. M. DeitelによるOperating Systems, Addison-Wesley (1984)の21章を参照されたい。

【0018】従来の2相コミット・プロトコル図5は、従来の2相コミット・プロトコルの代表例を示すタイミングチャートである。このタイミングチャートは、2つのリソース・マネジャP₁、P₂を想定している。ユーザはそのいずれかまたは両方を呼出すことができる。リソース・マネジャP₁、P₂は、DB2、IMS/DL1などのプログラムである。DB2はIBMデータベース製

品で、IMS/DL1はIBM情報管理システム製品であり、このトランザクション処理システムにDL/1データベースへのアクセスを与えるサービス・プログラムを含む。ただDB2とIMS/DL1は一例にすぎず、他のトランザクション処理システムも使用できる。コーディネータは図5ではCOORと呼ばれ、図1乃至図4に関して挙げたIBMのMVS (多重仮想記憶) オペレーティング・システムなど適切なオペレーティング・システムのモジュールでもある。

【0019】ここで図5と図8を参照する。この例のプロセスは図8の機能ブロック101から始まり、ここでユーザがデータを変更するかコミット/バックアウトするかを決定する。決定がデータの変更であれば、(この例では) 変更されるデータがP₁データかP₂データかがブロック102で判定される。P₁データなら、P₁データが機能ブロック103で変更される。これは図5ではステップ1として示した。データが変更されるとP₁が独自にこれらデータ変更結果をログする (図8の機能ブロック104)。次に図8の機能ブロック105と図5のステップ2で、P₁がコーディネータ (COOR) に、このユーザがそのデータ変更結果を何時コミットまたはバックアウトしようとするかの通知を受けなければならないことを伝える。ここで処理がユーザに戻る。

【0020】ここでユーザがあるP₂データを変更するとする。その場合、プロセスは判定ブロック101、102から機能ブロック106に進み、そこでP₂データが変更される。図5ではこれをステップ3として示した。P₁のデータ変更と同様にデータが変更された時、P₂は独自にこれらデータ変更結果をログする (図8の機能ブロック107)。次に図8の機能ブロック108と図5のステップ4で、P₂はコーディネータ (COOR) に、このユーザが何時そのデータ変更結果をコミットまたはバックアウトするかという通知を受けなければならないことを伝える。この点で処理はユーザに戻る。

【0021】2相コミット・プロセスは図5のステップ5から始まる。例ではこの時、ユーザが2つの加入者P₁、P₂でそのデータ変更結果をすべてコミットすることに決める。これは、適切なコマンドをコーディネータ (COOR) に対して直接実行することによって行なわれる (図5のステップ5)。コーディネータは、このコマンドを受信してから、データ変更結果の硬化を2つの加入者P₁、P₂と同期化しなければならない。2相コミット・プロトコルの第1ステップとして、コーディネータは機能ブロック110でPREPARE信号をP₂、P₁に送る (図5のステップ6、7)。コーディネータは次に、P₂、P₁からの投票を待つ (図8の判定ブロック111)。

【0022】図5でP₁はPREPARE信号にYESを投じる (ステップ8)。これは、P₁がデータ変更結果を硬化するか、またはバックアウトするかのいずれ

か、コーディネータが指示したとおりに準備が整っていることを示す。図5のステップ9でP₂も同様にPREPARE信号にYESを投じる。すなわちP₂はここで前または後に進む準備が出来ている。この時、コーディネータは加入者から期待していた投票をすべて受信している。コーディネータは、判定ブロック112で投票をもとにして（すべての投票がYESなら）前へ進むか、（すべての投票がYESでなければ）後へ進むかを決定する。これはアトミック・インスタントとして知られている。

【0023】この例では、いずれの投票もYESなので、コーディネータは前へ進み、レコードを保証された硬化媒体上のログに書き込むことを決定する（機能ブロック113）。図5ではこれをステップ10に“*”で示している。これは決定が行なわれたという意味である。ログに書き込みをすることによって、システムが障害を起こすか、またはたとえリソース・マネジャでも障害を起こした時は、コーディネータは再び同期をとり、各加入者に前または後へ進むかどうかを伝えることができる。再開時にログにアトミック・インスタント・レコードがある場合、コーディネータは加入者に、ログ・レコードに記録された決定事項に応じて前または後へ進むことを伝える。ログにアトミック・インスタント・レコードがない場合、決定が行なわれる前に障害が発生している。その場合の処理はすべてのデータ変更結果をバックアウトすることである。

【0024】図の例の場合、コーディネータは前進する（すなわちコミットする）という信号をP₂に送り（図5のステップ11）、図5のステップ12で後進するという信号をP₁に送る。このコミット機能は図8の機能ブロック114に示した。コーディネータは、前進することを加入者に通知した後、各加入者がそのデータ変更結果を硬化したことを確認応答するまで待たなければならない（図8の機能ブロック115）。図5ではP₁がコーディネータに、そのデータ変更結果の硬化を終えたことを通知し（ステップ13）、P₂はコーディネータに、そのデータ変更結果の硬化を終えたことをステップ14で通知する。これら確認応答が受信されると、コーディネータはユーザP₁とP₂の関係を忘れることができる。従って、コーディネータは別のログ・レコードに書き込みをし（図5のステップ15の“#”）、先に書き込みをしたアトミック・インスタント・レコードを取消す。これによりコーディネータは、再開時にログを読取る際に作業単位が終了したかどうかを知ることができる。図8の機能ブロック116でログが変更された後、処理はユーザに戻る。

【0025】図5、図8について説明したプロセスは従来の2相コミット・プロトコルの応用である。本発明の背景を説明するためにのみここに記した。加入者P₁、P₂の独自のログ処理の詳細は図5、図8には示してい

ない。ただし、これら“加入者”ログの詳細は分散コミットの流れを理解するうえで大切である。P₁、P₂のログは、図5のステップ10のコーディネータ・ログに示すように、アトミック・インスタント・ログ・レコードを含まない。このレコードを持つのはコーディネータだけである。加入者は代わりに他の一連のエントリ・ログを使用する。これらの加入者のログ・エントリの詳細は、重要ではあるが本発明には含まれない。

【0026】図5のタイミングチャートに示した基本的な2相コミット・プロトコルは、PREPARE信号に別の応答を与えることによって最適化されている。この応答はFORGETまたはREAD ONLYである。図6に、図5のタイミングチャートをもとにし、加入者P₁、P₂の両方がコーディネータからのPREPARE信号に応答してFORGETを投じる場合のタイミングチャートを示す。ステップ10乃至14は省略している。すなわちコーディネータは、FORGET応答をすべて受信した後、前後に進む決定はせず、記録されるアトミック・インスタントはなく、コーディネータはただユーザと加入者P₁、P₂の関係を忘れるだけである（ステップ15）。図7も図5のタイミングチャートをもとにし、P₁がコーディネータからのPREPARE信号にFORGETを投じるが、P₂はYESを投じる場合のタイミングチャートを示す。このケースでは加入者P₁に関係するステップ12、13を略しているが、他のステップはステップ10のアトミック・インスタントを含めてすべて実行される。図9は、図8の流れ図をもとにし、図6、図7に示した例についてREAD ONLY最適化をサポートする変更例を示す流れ図である。図9で判定ブロック112aは図8の判定ブロック112に代わる。すべての投票がYESかどうか判定されるのではなく、少なくとも1つの投票がYESで、NO投票がないという条件がチェックされる。つまり、1つの投票がFORGETである可能性はあるが（図7）、YES応答があるため、機能ブロック113でのログへの書き込みが続いて、機能ブロック114aでCOMMIT信号が、FORGETを投じた加入者を除く全加入者に送られる。一方、判定ブロック112aで判定された条件が見つからない場合、更に判定ブロック117で、全投票がFORGETかどうか判定される（このケースは図6）。FORGETの場合、プロセスは直接に機能ブロック116に進み、処理がユーザに戻る前にログが更新される。

【0027】従来の2相コミット・プロトコルは、分散システムにうまく適合しないことが知られている。先に挙げたDateの文献を参照されたい。問題はシステムがどのようなタイプの通信媒体とも接続可能なので（APPC-拡張プログラム間通信、TCP/IP-伝送制御プロトコル/インターネット・プロトコル、OSI/CSなど）、コーディネータの実行結果をシステム全体に行き

渡らさせるのは簡単ではないということである。コーディネータの役割を拡張し、加入者と通信するためのメカニズムから切り離さなければならない。これは本発明に従って、システム間に用いられる通信メカニズムに依存せずにコーディネータの役割を分散させることができる2相コミット・プロトコルの拡張機能によって行なわれる。

【0028】本発明に従った2相コミット・プロトコルの拡張：2相コミット・プロトコルに特別処理ステージが追加される。この特別処理ステージは第1相終了処理（EPOP）と呼ばれ、コーディネータ機能を任意の通信メカニズムを用いるシステム全体に分散させることができる。第1相終了処理は加入者が制御権を得られる特別ステージである。このステージで加入者は、2相コミット・プロトコルのシーケンスを分散システムに送る。通信メカニズムは、どのようなタイプのものでも、それが分散コーディネータの一部になるように用いられる。コーディネータそれ自体は他のシステムの知識を要しない。

【0029】本発明の実施例についての以下の説明に用いる用語は最初に定義しておく必要がある。新しい用語は（本発明に関係する場合は）「」で示す。

【0030】「PREPAREに対するABSTAIN」第1相で送られたPREPARE信号にABSTAINを返す加入者は、第1相の終わりに収集された投票に影響を与えない。ABSTAINを返す加入者には、第2相でCOMMITまたはCOMMIT信号が送られ、加入者が第1相終了処理を有効にした場合には第1相終了処理が駆動される。

【0031】「コーディネータのエージェント（AOC）」コーディネータの代理を務めるサブコーディネータ。AOCは、加入者の一部またはユーザの分散の度合いに応じて多くなる。AOCの主な役割は、分散したコーディネータ機能を実行することである。

【0032】「アトミック」アトミックは「ただ1つ」の意味で用いられる。一連のデータ変更がアトミックに行なわれるのは、すべて行なわれるか、または全く行なわれない場合である。ある変更を加えて別の変更を加えない状態は、それらがアトミックに行なわれた時には存在しない。

【0033】「アトミック・インスタント」データ変更結果をすべてコミットするか、またはデータ変更結果をすべてバックアウトするかの決定が成された時の時間点をいう。

【0034】「アトミック・インスタント・レコード（T₀ともいう）」コーディネータが決定をコミットまたはバックアウトしたことを示すために硬化媒体に書込まれるレコード。コーディネータしかこのレコードを書込めない。

【0035】「第2相開始ログ・レコード（B₀ともい

う）」AOCがコーディネータからコミットまたはバックアウトの決定を受信してログしたことを示すために硬化媒体に書込まれるレコード。これは、以下に定義するE₀と同様にAOCまたは加入者の活動である。

【0036】「コミット・スコープ」データの変更がすべて1つのシステムで行なわれた場合、コミット・スコープはローカルである。いくつかのデータ変更が2つ以上のシステムで行なわれ、ユーザがこれらの変更をアトミックに行なおうとした場合、コミット・スコープは分散する。

【0037】「コーディネータ」2相コミット・プロセスのコントローラ。分散プロセスでは1つのコーディネータしかない。

【0038】「分散」プロセスは、独立してはいないが関係のあるコンピュータ・システムに存在するいくつかのプロセスから成る場合は、分散している。

【0039】「第1相終了処理イネーブル（EEPOP）」ローカル・システムに、それがいまコーディネータのエージェント（AOC）であることを通知する（ローカル・コーディネータに通知する）新しいオペレーティング・システム・サービス・コール。すなわちローカル・コーディネータには、それが実際に、分散したコミット・スコープの一部でしかないことが伝えられる。コミット・スコープにはコーディネータが1つしかなく、このサービスにより、ローカル・オペレーティング・システムは、コーディネータが別のシステムに存在し、ローカル・システムがそのコーディネータのエージェントであることを認識する。このサービスはローカル・オペレーティング・システムをコーディネータからAOCに変換する。

【0040】「第1相終了ログ・レコード（E₀ともいう）」2相コミットの第1相が完了したことを示すために硬化媒体に書込まれるレコード。これはAOCまたは加入者の活動である。すなわちこのレコードがコーディネータによって書込まれることはない。これは、AOCまたは加入者がコーディネータまたはコーディネータと信じられるエージェントからのPREPARE信号（別のAOCから来ることもある）に回答して投票したことを示す。

【0041】「第1相終了処理（EPOP）」第1相終了レコードがログに書込まれた後に生起する新しい処理ステージ。これは、ローカル加入者がすべて投票を終えたことを示す。更に投票（コミットまたはバックアウト）の結果も示す。これは2相コミット・プロセスを保留し、加入者が投票の結果を分散システムに通信できるようにし、これにより分散コミット・スコープの一部になることができる。

【0042】「ログ」重要データが格納された硬化媒体（通常は磁気ディスク装置）。

【0043】「加入者」2相コミット・プロセスの「コ

ントローリ”(被制御装置)。加入者は、ユーザが使用しているリソース(データベース内のデータや別ユーザへの通信バスなど)を所有する。ユーザは加入者をいくつでも関係させることができる。加入者は分散させることができる。

【0044】「第1相(2相コミットの)」VOTE(投票)相ともいう。これはPREPARE信号を加入者に送る相から成る。

【0045】「第2相(2相コミットの)」ACTION(実行)相ともいう。これはCOMMITまたはBACKOUT信号を加入者に送る相から成る。

【0046】「プロセス」プロセスはコンピュータ・システム内の作業単位であり、ユーザ、プログラム製品、単一スレッド、複数のスレッドなどを含む。

【0047】「ユーザ」いくつかのシステムに分散しているか分散していないアプリケーション・プログラム。

【0048】「読取り専用最適化」2相コミット・プロトコルに対する最適化。加入者は第1相で送られたPREPARE信号にFORGETまたはREAD ONLYを返す。FORGETを返す加入者には、第2相でCOMMITまたはBACKOUT信号は送られない。コミット・スコープの全加入者がPREPARE信号にFORGETを返した場合、アトミック・インスタントはログされず、第1相終了(E₀₁)または第2相開始(B₀₂)のレコードを実行する必要がある。

【0049】図10にマルチプロセッサ・システムを示す。第1コンピュータ51(マシンA)は通信装置53を介して第2コンピュータ52(マシンB)に接続される。装置53は、銅、光ファイバ・ケーブルなど適切な通信リンクの通信媒体を含むことができる。コンピュータ51、52はそれぞれ図1乃至図4のコンピュータ・ライクなコンピュータ11とみることもでき、従って図1乃至図4の詳細はここでは簡単のため省略しているので、図10のブロック図はハイレベルなブロック図である。

【0050】図10に示した2マシン・システムは、本発明に従った分散システムとは何かを理解するうえで役立つ基礎概念を示すが、分散システムを1つのマシン上で実現することも可能である。たとえばIBMの仮想マシン(VM)オペレーティング・システム、VM/370はIBMシステム370コンピュータを管理し、端末から操作する数人のユーザそれぞれが完全なシステム370を持っているかのように見せる。また、各ユーザは異なるオペレーティング・システム(OS)を選択することができる。つまり、VMは一度に複数のオペレーティング・システムを(その仮想マシン上で1つを)実行することができる。VMの詳細については先に挙げたDitelの文献の第22章を参照されたい。

【0051】以下の説明では、マシンAとマシンBは物理的に独立したマシン(すなわち実マシン)とみるこ

ができ、或いは、仮想マシン・オペレーティング・システム上で走る1つのコンピュータ上の仮想マシンでもよい。もちろん図10よりも複雑なシステムでは、実マシンと仮想マシンを組合わせて、本発明に従った分散コミット・プロトコルが実現できる分散システムをサポートすることができる。

【0052】図10を参照する。コンピュータ51は、ユーザA、通信マネジャ(すなわちCMGR_A)、及びリソース・マネジャRM(すなわち加入者P₁)をサポートするオペレーティング・システム(OS)54を実行する。ユーザAのアドレス空間55、P₁のアドレス空間56、及びCMGR_Aのアドレス空間57はすべて、OS54と通信する。コンピュータ51には、それぞれCMGR_A、OS、及びP₁のログが書込まれるDASD58、59、60が接続される。同様にコンピュータ52は、第2ユーザB、通信マネジャ(すなわちCMGR_B)、及びリソース・マネジャRM(すなわち加入者P₂)をサポートするOS61を実行する。ユーザBのアドレス空間62、P₂のアドレス空間63、及びCMGR_Bのアドレス空間64はすべてOS61と通信する。コンピュータ52には、それぞれCMGR_B、OS61、及びP₂のログが書込まれるDASD65、66、67が接続される。各システム上に示した3つのログは同じ物理記憶装置でも異なる物理記憶装置でもよい。

【0053】動作時、マシンAのユーザAはCMGR_Aと対話し(破線矢印68)、それ自体の分散した部分をマシンBで実行するよう要求する。CMGR_Aはオペレーティング・システム54に、ユーザAが何時コミットまたはバックアウトするか通知を受ける必要があると通知する。これは破線69で示した。CMGR_Aは次に、通信装置またはリンク53を介してマシンB上のCMGR_Bと通信する。CMGR_AとCMGR_Bのいずれかまたは両方は、このトランザクションでプレイベート・データをそれぞれDASD58、65にログすることができる。次にCMGR_BはマシンBでユーザBプログラムを起動し(起動されていなかった場合)、オペレーティング・システム61に、ユーザBが何時コミットまたはバックアウトしたか通知を受ける必要があることを伝える(破線矢印70)。マシンAのユーザAとマシンBのユーザBは、APPCを用いる際には必要に応じて逆にしてもよい。

【0054】2つのマシンで処理されるトランザクションの一例として、マシンBのユーザBがデータベースP₂のローカル・データを変更することを破線矢印71に示す。その時、P₂はオペレーティング・システム61に、マシンAのユーザAまたはマシンBのユーザBが何時コミットまたはバックアウトするかの通知を受ける必要があることを伝える(破線矢印72)。破線矢印73は、マシンAのユーザAが次にデータベースP₁のロー

カル・データを変更することを示す。破線矢印73、71で示したデータベースP₁、P₂の変更は任意の順序で繰返し生じ得る。破線矢印75はユーザAがすべての変更結果をコミットしたいことを示す。

【0055】図11乃至図13を参照する。図11はユーザ、コーディネータ(COOR)、加入者(P₁、P₂)、コーディネータのエージェント(AOC)、及び通信マネージャ(CMGR)の役割を、図10のマシン例について示す。加入者P₁、P₂は、たとえば図5乃至図8の例のようにDB2、IMS/DL1などのデータベースである。ただし本発明はどのような用途にも限定されない。CMGRは実際には加入者にすぎないが、コーディネータ及びAOCと協同して分散コミット・プロセスを実現するので区別される。図12、図13は、EPOPサービスによって有効になった時に行なえる新しいオペレーティング・システム・サービス(EEPOEP)と終了処理を示す。

【0056】図の例でユーザは分散している。プロセスは、データをリソース・マネージャ(加入者P₁など)で変更するマシンA上で実行される。また、データをリソース・マネージャ(加入者P₂など)で変更するマシンB上でもプロセスが実行される。ユーザは次に、すべてのデータ変更結果をコミットするよう要求する。両システムのデータ変更結果はアトミックにコミットされなければならない。ユーザは、その処理を通信マネージャCMGR(APPC、TCP/IP、OSI/CSなどの通信マネージャ)で分散することに決める。用いられるメカニズムによって、ユーザはコミット・プロセスをコーディネータからか、または通信マネージャから直接要求する。EPOPは、ユーザが何らかの手段によってコミットしたいという希望をコーディネータが知らされる場合は、これに依存しない。

【0057】上記のとおり、ユーザは図11乃至図13の例ではそれ自体をすでに分散している。すなわち関係のある2つのマシン上でプロセスを実行している。ここでも、これらのマシンは実マシン(図10)か、仮想マシンすなわち仮想マシン・オペレーティング・システムにより1つのコンピュータ上で動作するマシンである。図11乃至図13には、処理の流れのうちコミットの部分しか示していない。これが本発明の新規性の元になる部分である。また加入者(P₁、P₂、CMGR_A、及びCMGR_B)はすべて、ユーザが何時データ変更結果すべてをコミットまたはバックアウトすることを決定したかの通知を受けたいとコーディネータまたはAOCに伝えている。実現形態によるが、図11のタイミングチャートに示した時点よりも後に、処理が正常終了したこと(すなわち変更結果がすべてバックアウトされたこと)を示す戻りコードをユーザが受取ることがある。図に示したログ・レコードはすべて、オペレーティング・システムが所有するログにある。加入者は、プライベ

ト・ログを持つこともできるが、これらプライベート・ログは図示していない。

【0058】図12のプロセスはマシンAで始まり、判定ブロック201で、ユーザが変更結果をコミットしたかどうか検出される。図11のステップ1では、ユーザがデータ変更結果をすべてコミットしたことを示す。図12でコーディネータは、PREPARE信号を加入者P₁と通信マネージャCMGR_Aの両方に同報する(機能ブロック202)。これは図11のステップ2である。図12でCMGR_Aは、図10の通信68から、分散コミット・シーケンスで別のシステムと通信していることを認識し、機能ブロック203(図11のステップ4)でPREPARE信号をCMGR_Bに送る。これは、用いられている通信マネージャに依存しない。

【0059】図13で、プロセスは判定ブロック204のマシンBから始まり、CMGR_Aからのメッセージが検出される。判定ブロック205でPREPARE信号が検出される。CMGR_Bは、PREPARE信号を受取った時、これ(すなわちマシンB)がコーディネータのホームではないことを認識する。これが分かるのはPREPARE信号を受信したからである。これはコーディネータの"領域"ではないので、コーディネータのエージェント(AOC)でなければならない。図11のステップ5を参照されたい。CMGR_Bはそこで第1相終了処理イネーブル(EEPOEP)と呼ばれる新しいオペレーティング・システム・サービスを機能ブロック206で実行する。これでオペレーティング・システムに、これがAOCであることが通知される。またこれにより、第1相終了ログ・レコード(E₀₁)が書込まれた後(図11のステップ10の後)制御がCMGR_Bに渡る。次に処理はCMGR_Bに戻る。

【0060】ここでAOCは、PREPARE信号をP₂とCMGR_Bに同報する(機能ブロック207)。これは、(1)ユーザがコミット・リクエストを実行することを通信マネージャによって指示されたか、または(2)AOCが、新しいEEPOEPサービス・コールを受信した時がコミット・プロセスを開始する時と仮定したことのいずれかに応じて行なわれる。図11のステップ6を参照されたい。図11に示した例の場合、CMGR_Bはステップ7でPREPARE信号にYESを投じ、P₂は、ステップ8でPREPARE信号にYESを投じる。これらの投票は図13では判定ブロック208で検出される。

【0061】次にAOCは、(1)その投票をすべての加入者(CMGR_BとP₂)から受信し、(2)図11のステップ5で実行されたEEPOEPサービスによって、それがAOCであることがこのオペレーティング・システム・インスタンスに通知されたので、機能ブロック209で第1相終了(E₀₁)レコードをログに書込

む。このレコードを書込むのはAOCだけである。このレコードは図11のステップ9のタイミングチャートに"*"で示した。次にAOCは、CMGR_Bが先に第1相終了処理イネーブル(EPEOP)サービスを実行していたので、ステップ10で制御をCMGR_Bに渡す。これは、第1相終了と呼ばれる新しい処理ステージである。CMGR_Bは、PREPARE信号の結果を通知される。第1相終了では、CMGR_Bが次にCMGR_AにYESを返す(図13の機能ブロック210)。図11のステップ11を参照されたい。これは、ステップ4でCMGR_AがCMGR_Bに送ったPREPAREへの応答である。図の例では、すべての投票がYESであり、PREPAREの結果はYESだった(すなわちコミット処理に進む)。PREPARE信号は他の結果もあり得るが、本発明の説明を補足するものではないので省略する。

【0062】図11のステップ3でP₁は、それが準備を選択することを示し(この例ではYES)、これは、それが行なったデータ変更の結果をすべてコミットまたはバックアウトする準備が出来ていることを示す。P₁による投票とCMGR_Bによって返った投票は、図12の判定ブロック211で検出される。CMGR_Aはまた、ステップ2でそれにローカルに送られたPREPARE信号にYESを投じる。これは、全加入者(他のマシンに分散することもある)がいまPREPARED STATEにあることを認識して行なう。この知識はステップ11から得られる。すなわち全加入者が前か後に進む準備が出来ている。これは図11のステップ12で生じる。

【0063】マシンAのコーディネータはここで、それが予測していたすべての投票を受取っており、この例では全投票がYESである(判定ブロック212)。そこでコーディネータは、前進してアトミック・インスタント・レコード(T₀₂)をログに書込むことを決定する(機能ブロック213)。このT₀₂レコードは、図11のステップ13に"*"で示した。コーディネータはステップ14でコミット・オーダをローカル加入者P₁とCMGR_Aに同報する。これは図12では機能ブロック214に示した。CMGR_Aは、コミット・オーダを受信した時、コミット・オーダをCMGR_Bに送る(機能ブロック215)。これは図11のステップ15である。

【0064】図13で、CMGR_Aからのメッセージが再び判定ブロック204で検出される。ただこの時のメッセージはCOMMITである(判定ブロック205)。CMGR_Bは、決定(例ではYESなど)をAOCに返すことによってその第1相終了処理を終えている。図11ではこれはステップ17である。ここでAOCは第1相終了処理が完了し、コーディネータ領域からの決定は前進(すなわちコミット)であることを認識す

る。AOCは第2相開始(B₀₂)レコードをローカル・ログに書込む(図13の機能ブロック216)。復元シナリオのこの時点から以降、ローカル・システムはコーディネータとの通信を再確立してコミット/バックアウト決定を得る必要がない。決定はB₀₂ログ・レコードに記録される。このレコードが書込まれるのは、ローカル・システムがEPEOPサービスによってAOCになったからである。B₀₂レコードは、コーディネータによって書込まれることはなく、AOCだけが書込む。このB₀₂レコードは図11では"*"で示した。

【0065】AOCは、コミット・オーダをマシンBのローカル加入者P₂とCMGR_Bに同報する(図13の機能ブロック217)。図11ではこれをステップ19に示した。AOCは、ログ・レコードの書込みと第1相終了処理(EPOP)以外はコーディネータと同じように機能する。CMGR_Bは機能ブロック218でCMGR_Aに、データ変更結果のコミットを終えたこと(すなわちB₀₂レコードがログ上で安全であること)を通知する(図11のステップ20)。CMGR_Bは図11のステップ22で、ステップ29のコミット同報からAOCに戻る。これによりAOCは、CMGR_Bがそのコミット処理を終えたとの通知を受ける。またP₂は、ある点で(図11のステップ23など)ステップ19のコミット同報からAOCに戻る。これは図13の判定ブロック219で検出され、CMGR_BとP₂がデータ変更結果のコミットを終えたことがAOCに通知される。AOCは、全加入者がそのデータ変更を終えたことを認識した後はいつでも(図11のステップ25など)、これを機能ブロック220でログに記録する。これは性能の改良であり、再開処理を高速化する。マシンBのAOCは、それが分散コミット・スコープに加わったことを認識していたが、用いられた通信メカニズムまたはプロトコルは認識しなかった。書込まれたレコードは図11のステップ25では"X"で示した。

【0066】P₁は、ある点で(図11のステップ16)、それがデータ変更結果のコミットを終えたことをマシンAのコーディネータに通知する。図11のステップ21でCMGR_Aもステップ14のコミット同報からコーディネータに戻る。これは、図12の判定ブロック221で検出される。全加入者がそのデータ変更を終えたことをコーディネータが認識した後はいつでも(図11のステップ24など)、コーディネータはこれをログに記録する(機能ブロック222)。これは性能の改良であり再開処理を高速化する。

【0067】コーディネータは、それが分散コミット・スコープに加わったことを認識していなかった。このレコードは図11のステップ24には"X"で示した。図11の例のキーポイントは、処理シーケンスを示し、データが常に調整されるようにすることである。すなわちユーザのデータの一部だけを更新することは常に許されな

い。変更はすべて要求に応じて行なわれるか、または、何か問題がある場合はすべての変更が逆にされる。この新しいステップ (EPOPとEEPOEP) はデータの保全性を保つ。図11の例では、前進する判定が成された。バックアウト判定も行なえる。これもログに記録され同様の処理が続く。

【0068】図11に示した例の処理により、障害が発生した場合にはいつでもすべてのデータが調整されるようになる。これは、データの一意性が保たれると表現されることもある。

【0069】ここで説明を簡単にするために、障害は図11に示した分散システムの両方が共に障害を起こしたものとする。これより簡単なケースで、システムが1つだけ障害を起こした場合は、両方が同時に障害を起こした最悪のシナリオに含まれる。同様に、両方が故障したが同時にではない場合も、用いられているログ・レコードが特定の時間順で書込まれるという事実により扱える。また、分散コミット・スコープに参加している分散システムが2つ以上の場合も同様である。これと同じロジックは、高さが3以上、幅が2以上のツリーに展開される。図11の例では、ツリーの高さは2で、親であるマシンAの子は1つ、幅は1になる。

【0070】下記の障害例では図11を参照する。

【0071】例1: E₀₁ ログ・レコードがログに書込まれる前に障害が発生する (すなわちAとした時間より前)。再開時に、マシンAで、P₁ がそのプライベート・ログ (図示なし) を読取り、ユーザのためにデータを変更していたことを発見する。P₁ は、オペレーティング・システムにコミットまたはバックアウトの決定を依頼する。オペレーティング・システム (すなわちコーディネータ) は、"ユーザ"に関係するそのログでT₀₂またはB₀₂を発見せず、これをP₁に通知する。P₁はそこですべてのデータ変更結果をバックアウトする。CMGR_Aはそのプライベート・ログを読取り、それがユーザのためにデータを変更していたことを発見する。CMGR_Aはオペレーティング・システムにコミットまたはバックアウトの決定を依頼する。オペレーティング・システム (すなわちコーディネータ) は、ユーザに関するそのログでT₀₂またはB₀₂を見つせず、これをCMGR_Aに通知する。CMGR_Aは次に、ある場合はすべてのデータ変更結果をバックアウトするか、またはこのユーザを無視する。マシンBで、P₂ はマシンAの場合と同様、このステップの処理を同じ実行結果で行なう。CMGR_Aは、マシンAの場合と同様、このステップの処理を同じ実行結果で行なう。

【0072】例2: T₀₂ ログ・レコードが時間Bに書込まれる前の時間AにE₀₁ ログ・レコードが書込まれた後に障害が発生する。再開時、マシンAで、P₁ は例1のように処理を実行する。CMGR_Aは例1のように処理を実行する。マシンBで、P₂ は、例1のように処理

を実行し、オペレーティング・システムによって、ユーザの状態がIN_DOUBTであると通知される。P₂ はこれを、コーディネータが (別のマシンで) T₀₂レコードを書込んだかどうかをローカル・システムがまだ認識していないと解釈する。すなわちコーディネータが決定をコミットまたはバックアウトしたかどうか認識しない。その後P₂ は通知を受ける。すなわちコーディネータとの再同期化が完了した後である。P₂ は、それがまだすべての変更をコミットまたはバックアウトする準備を整えた状態のままでなければならないことを覚えておく必要がある。CMGR_Aは例1のように処理を実行し、オペレーティング・システムによって、ユーザの状態がIN_DOUBTであると通知される。CMGR_Aは、それがログに格納した情報から、コーディネータ (またはおそらくは単に他のAOC) がマシンAにあることを認識する。そのログから、CMGR_Aとの通信を再確立するのに十分な情報を読取る。次にCMGR_Aに、コミットまたはバックアウトの判定を求める。CMGR_Aは、それが再開操作を終えている場合は、T₀₂またはB₀₂ログ・レコードが見つからなかったことを認識する。CMGR_AはCMGR_Bに、判定がバックアウトであることを通知する。CMGR_BはAOCにこれを通知し、AOCは、対象加入者に判定がバックアウトであることを通知する。例でAOCはP₂に通知する。

【0073】例3: B₀₂ ログ・レコードが時間Cに書込まれる前に (時間Bなどに) T₀₂ ログ・レコードが書込まれた後で障害が発生する。再開時、マシンAで、P₁ は例1のように処理を実行し、ユーザが状態IN_COMMITにあることをオペレーティング・システムによって通知される。P₁ はこれを、ユーザのためのデータ変更をコミットしなければならないと解釈し、そう実行する。CMGR_Aは、例1のように処理を実行し、ユーザが状態IN_COMMITにあるという通知を受ける。CMGR_Aはこの情報を、(1) CMGR_AがCMGR_Bと再同期化するか、または(2) CMGR_BがCMGR_Aにユーザの判定を求めた時に、CMGR_Aに引渡す。CMGR_AはこれをAOCに通知し、AOCはそこで、すべての対象加入者に判定がコミットであることを通知する。例でAOCはP₂に通知する。マシンBで、P₂ は、例2のように処理を実行する。CMGR_Bは例2のように処理を実行するが、T₀₂レコードが見つかったことをCMGR_AがCMGR_Bに通知する。従って、判定はすべての変更をコミットすることである。CMGR_BはこれをAOCに通知し、AOCはそこですべての対象加入者 (例ではP₂) に通知する。

【0074】例4: 最終レコードのいずれか (図11のステップ24、25) が書込まれる前にB₀₂レコードが (時間Cで) 書込まれた後に障害が発生する。その場合

P₁、またはP₂は例1のように処理を実行し、ユーザの状態がIN_COMMITであることをオペレーティング・システムから通知される。従ってP₁、P₂はすべてのデータ変更結果をコミットする。一方、CMGR_AとCMGR_Bは例1のように処理を実行し、ユーザの状態がIN_COMMITであることをオペレーティング・システムから通知される。用いられているプロトコルによるが、CMGR_AとCMGR_Bは、それらが再同期化した時にこの情報を交換する。ただし、コミットまたはバックアウトの判定は、各システムで使用できるログ・レコードから個別に推定できるので、再同期化時に交換された情報は用いられない。

【0075】例5：最終レコードのいずれかまたは両方が時間D、Eで書込まれた後に障害が発生する。その場合P₁、P₂は、(1)ユーザの状態がIN_COMMITであることをオペレーティング・システムから通知され、すべてのデータ変更結果をコミットするか、または(2)ユーザの代わりに行なわれたすべての作業がすでに終わっており、何の実行も必要ないことが通知される(または推定することができる)。一方、CMGR_AとCMGR_Bは、(1)ユーザの状態がIN_COMMITであることをオペレーティング・システムから通知され、例4のように処理を実行するか、または(2)ユーザの代わりに行なわれたすべての作業がすでに終わっており、何の実行も必要ないことが通知される(または推定することができる)。

【0076】以上、復元フローについて説明したが、これは加入者がここに示したプロシジャに従う際、データの安全性が常に保たれることを示すものである。それ以上のデータ安全性は、標準的な2相コミットについては明らかにされていない。EEPOEPサービスと第1相終了処理により、分散コミット・スコープが有効になり、分散コーディネータが必要でなくなる。すなわちコーディネータは、特定の通信プロトコルまたは分散ネットワーク・トポロジを認識もせず、それに対応もしなかったということである。EEPOEPサービスでコーディネータのエージェント(AOC)領域を形成することによって、システムの数にかかわらずにコミット・スコープの分散を維持できる。例では2つのシステムしかコミット・スコープに参加していないが、この機構を用いることで、参加できるシステムの数は制限されない。

【0077】コーディネータの機能はデフォルトであるが、AOCの機能は、新しいオペレーティング・システム・コール(すなわちEEPOEP)1つで実現できる。それでもシステム間通信の流れは標準的な2相コミットである。すなわち例では、マシンAとマシンB間の流れは拡張機能がなく2相コミットとみられる。コーディネータ・システムはなお標準的な2相コミットとみられる。すなわち例では、マシンAの流れは標準的とみられる。マシンAのコーディネータは、コミット・スコ

プが分散していることを認識させない。適切な時間にEEPOEPサービスによって、マシンBがコーディネータからコーディネータのエージェント(AOC)に変換される。

【0078】上記の内容から明らかなように、本発明は、コーディネータを効率よく分散させるが、コーディネータはこの事実の知識をもっている必要はなく、大幅に簡素化される。コーディネータ機能は、任意の通信メカニズムを使用して分散される。ユーザは分散コーディネータ機能を意識する必要がない。更に、汎用リソース・マネージャは、それに合った通信マネージャを使い、この新しいプロトコル拡張機能を認識し、オペレーティング・システム同期点マネージャ(すなわちコーディネータ)と通信することによって分散させることができる。この方法の新規性は、コーディネータ機能が通信媒体に依存せずに分散されることである。コーディネータにより、特定の通信製品の知識がないまま分散コミット・スコープ(すなわち拡張2相コミット)が有効になる。第1相終了処理という拡張機能では、この分散を、分散しているコーディネータ機能をオペレーティング・システムが認識しないままに実現することができる。

【0079】このほか、分散加入者に対する2相コミット・プロトコルの拡張機能として、コーディネータからのPREPARE信号に対してリソース・マネージャが新しい応答を使用することができる。この応答は、PREPARE信号に回答してコーディネータのエージェントに回答するために通信リソース・マネージャ(CMGR_A)が使用できるABSTAIN応答である。この新しい応答により、性能向上、システム・スループットの向上、及び分散システム上の再開時間の短縮が可能になる。

【0080】図14に、全加入者がマシンBでFORGETまたはABSTAINを、マシンAでFORGETを選択する場合について、分散2相コミットのタイミングチャートを示す。図14は図11をもとにしており、ステップはいくつか削除している。

【0081】従来の2相コミット・プロトコルでは、リソース・マネージャがコーディネータからのPREPAREリクエストにある方法で応答する必要がある。加入者はそのリクエストに対して、コミット・コーディネータへのYES/COMMIT、NO/BACKOUT、またはFORGET/READ ONLYで応答しなければならない。しかし、コーディネータのエージェントが、YES/COMMIT、NO/BACKOUT、またはFORGET/READ ONLYの応答を可能にするだけであれば、読取り専用最適化を行なうことはできない。これは、CMGR_Bが図11のステップ7でPREPAREにYESと応答しなければならないという事実による。これによりAOC_Bは、更新されたコミット可能なリソースがマシンBにあるとみなし、従っ

て、加入者P₂がPREPARE信号にFORGET/READ ONLYを返した場合でも、2相コミット・プロセスを完全に実行しなければならない。図14のステップ7では、CMGR_BがAOC_BからのPREPARE信号にABSTAINを選択する。CMGR_BからのこのABSTAIN投票により、図10の次のステップは生じない。

1. ステップ9でAOC_Bによって第1相終了(E₀₁)がログに書込まれる。
2. ステップ11で通信応答YESがマシンAに送られる。
3. ステップ15で通信応答COMMITオーダがCMGR_Bに送られる。
4. ステップ18でマシンBのAOC_Bによって第2相開始(B₀₂)が書込まれる。
5. ステップ19、22、23のCOMMIT信号と応答。

これらのイベントはすべて、マシンBが実際に、図11の加入者P₂のように、2相コミット・プロトコル・リソースを持っている時に必要である。

【0082】加入者P₂が作業単位に関係しないか、加入者P₂しか読取り専用リソースを持っていない場合、これらのステップすべてが必要なわけではない。こうした状況で性能を向上させる2相コミット・プロセスのREAD ONLY最適化がある。図14のタイミングチャートはREAD ONLY最適化を示す。不要なプロセスは上記のログへの書込みと通信イベントなどである。CMGR_Bはステップ7でPREPARE信号にNOを返すことができない。これにより作業単位がバックアウトされる。CMGR_Bは、CMGR_Bが作業単位に関わろうとしなくなったとAOC_Bがみなすことになるので、FORGETを返すことができない。

【0083】ABSTAINは、本発明によって提供される新しい応答であり、CMGR_BはこれをPREPARE信号への応答に使用することができる。この応答は、ABSTAINで応答する加入者(例ではCMGR_B)が、PREPAREリクエストの最終結果に影響を与えようとしなく、どのような結果(YES、NO、またはFORGET)も受入れようとする、及び、現在の作業単位の2相コミットに関係し続けようとする、を示す。図14のステップ7では、CMGR_Bは、PREPARE信号にYESと答えるのではなく、ABSTAINを返す。加入者P₂がPREPARE信号にYESと答えた場合、AOC_BのPREPARE信号の最終結果はYESであり、イベントは上記のように進む。加入者P₂がFORGETと答えるか、加入者P₂が存在しない場合、AOC_B PREPARE信号の最終結果はFORGETである。AOC_Bは、図14のステップ10でFORGETの結果をCMGR_Bの第1相終了処理に返す。CMGR_Bはここ

で、ステップ11でのYES投票ではなく、通信応答FORGETをマシンA送る(ステップ20)。これによりマシンAは、作業単位についてマシンBと接触しなくなる。CMGR_Bは次にAOC_Bに戻り、AOC_Bはステップ25で"単位復元終了"レコードをログに書込み、作業単位についてのその処理を終える。

【0084】図14のタイミングチャートを図11と比較すると、マシンBに読取り専用リソースがある場合、マシンBの以下のステップはなくなる。

- 9) 第1相終了(E₀₁)をログする。
- 11) PREPAREに対してマシンAにYES応答を返す。
- 15) マシンAからコミット・オーダを通信する。
- 18) 第1相開始(B₀₂)をログする。
- 19) コミット・オーダをマシンBの加入者に引渡す。
- 22) CMGR_Bのコミット終了をAOC_Bに返す。

ステップ20はステップ10の後、ステップ12の前に移動する。以下のマシンA上のイベントは、マシンBが読取り専用リソースを持っている時にはなくなる。

- 14) コーディネータからCMGR_Aへコミット終了コールを出す。
- 21) CMGR_Aのコミット終了から復帰する。

【0085】図16は、図12の流れ図にもとづく流れ図で、図14のタイミングチャートに示したマシンAでの操作の変更を示す。判定ブロック212aで、YES投票が1つ以上でNO投票がないという条件がチェックされる。これはもちろん図9の判定ブロック112aで行なわれるチェックと同様である。また機能ブロック215はなくなり、機能ブロック214aは変更され、COMMITが加入者P₁にしか送られない。これは例でCMGR_AがFORGETを選択したからである。

【0086】図17は、図13の流れ図にもとづく流れ図で、図14のタイミングチャートに示したマシンBにおける操作の変更を示す。この流れ図で判定ブロック205と機能ブロック209、210、216、217はなくなっている。これらのブロックの代わりに新しい判定ブロック224がある。ここで受信された投票がすべてFORGETかABSTAINかが判定される。その場合、プロセスは機能ブロック218に進み、ACK信号がCMGR_Aに送られる。他の場合は処理がユーザに戻る。

【0087】マシンA、Bが、コミット対象の更新されたリソース(すなわち読取り専用)を持たない場合、マシンAから削除できるイベントがある。

【0088】13) "アトミック・インスタント"コミット・レコードT₀₂の強制ログ書込み、これにより性能が更に向上する。図15のタイミングチャートにこのイベントを示す。図18は、図12の流れ図にもとづく流れ図で、図15のタイミングチャートに示すマシンAにお

ける操作の変更を示す。判定ブロック212bは、すべての投票がFORGETかABSTAINかをチェックし、機能ブロック213、214、215、及び判定ブロック221はなくなっている。マシンBにおける操作は、図15に示したタイミングチャートについての図17の流れ図と同じである。

【0089】本発明は、分散データベースに、或いはマルチプロセッサ・システムにも限定されない。どのようなリソース・マネージャでも、或いはユーザでも、1つのマシン或いは任意個数のマシン上で分散させることができ、1つの大きなコミット・スコープに参加することができる。

【図面の簡単な説明】

【図1】本発明を実現できるトランザクション処理システムのブロック図である。

【図2】本発明を実現できるトランザクション処理システムのブロック図である。

【図3】本発明を実現できるトランザクション処理システムのブロック図である。

【図4】本発明を実現できるトランザクション処理システムのブロック図である。

【図5】現在の2相コミット・プロトコルの基本操作を示すタイミングチャートである。

【図6】2相コミット・プロトコルのREAD ONLY最適化の操作を示すタイミングチャートである。

【図7】2相コミット・プロトコルのREAD ONLY最適化の操作を示すタイミングチャートである。

【図8】図5に示した現在の基本的な2相コミット・プロトコルの流れ図である。

【図9】図6、図7に示したREAD ONLY最適化を実現する2相コミット・プロトコル・プロセスの流れ図である。

【図10】本発明に従った拡張2相コミット・プロトコルの操作を説明するのに役立つマルチプロセッサ・システムのブロック図である。

【図11】本発明に従った、図10の2マシン環境における2相コミット・プロトコルの拡張機能の操作を示すタイミングチャートである。

【図12】2相コミット・プロトコル・プロセスの拡張機能について、マシンAにおけるプロセスを示す流れ図である。

【図13】2相コミット・プロトコル・プロセスの拡張機能について、マシンBにおけるプロセスを示す流れ図である。

【図14】マシンBにおいて投票がすべてFORGETまたはABSTAINであり、マシンAにおいてCMGR_AがFORGETを選択した場合について、READ ONLY最適化と新しいABSTAIN応答をサポート

トする2相コミット・プロトコルの拡張操作を示すタイミングチャートである。

【図15】マシンA、Bの両方において投票がすべてFORGETまたはABSTAINである場合について、READ ONLY最適化と新しいABSTAIN応答をサポートする2相コミット・プロトコルの拡張操作を示すタイミングチャートである。

【図16】図14に示した例について、2相コミット・プロトコル・プロセスの拡張機能を改良するマシンAにおけるプロセスの流れ図である。

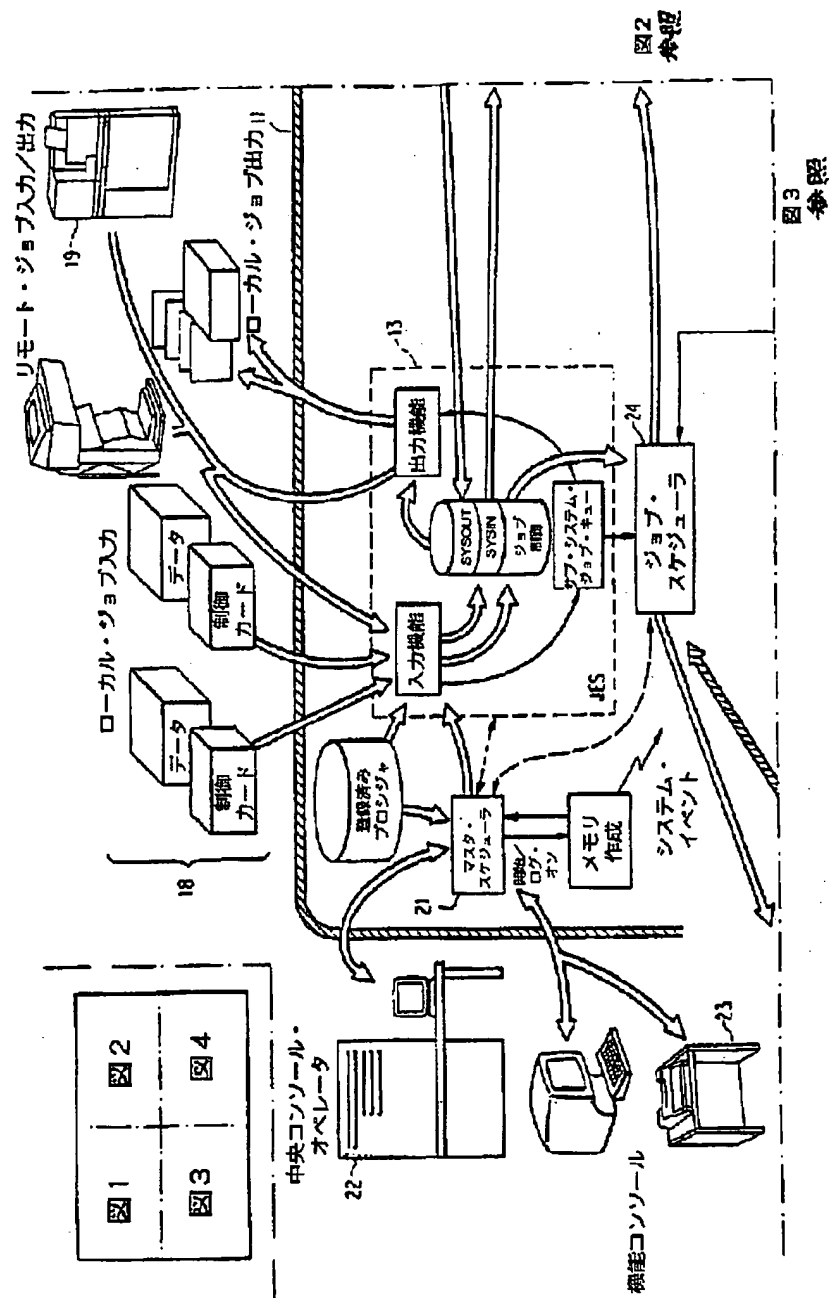
【図17】図14に示した例について、2相コミット・プロトコル・プロセスの拡張機能を改良するマシンBにおけるプロセスの流れ図である。

【図18】図15に示した例について、2相コミット・プロトコル・プロセスの拡張機能を改良するマシンAにおけるプロセスの流れ図である。

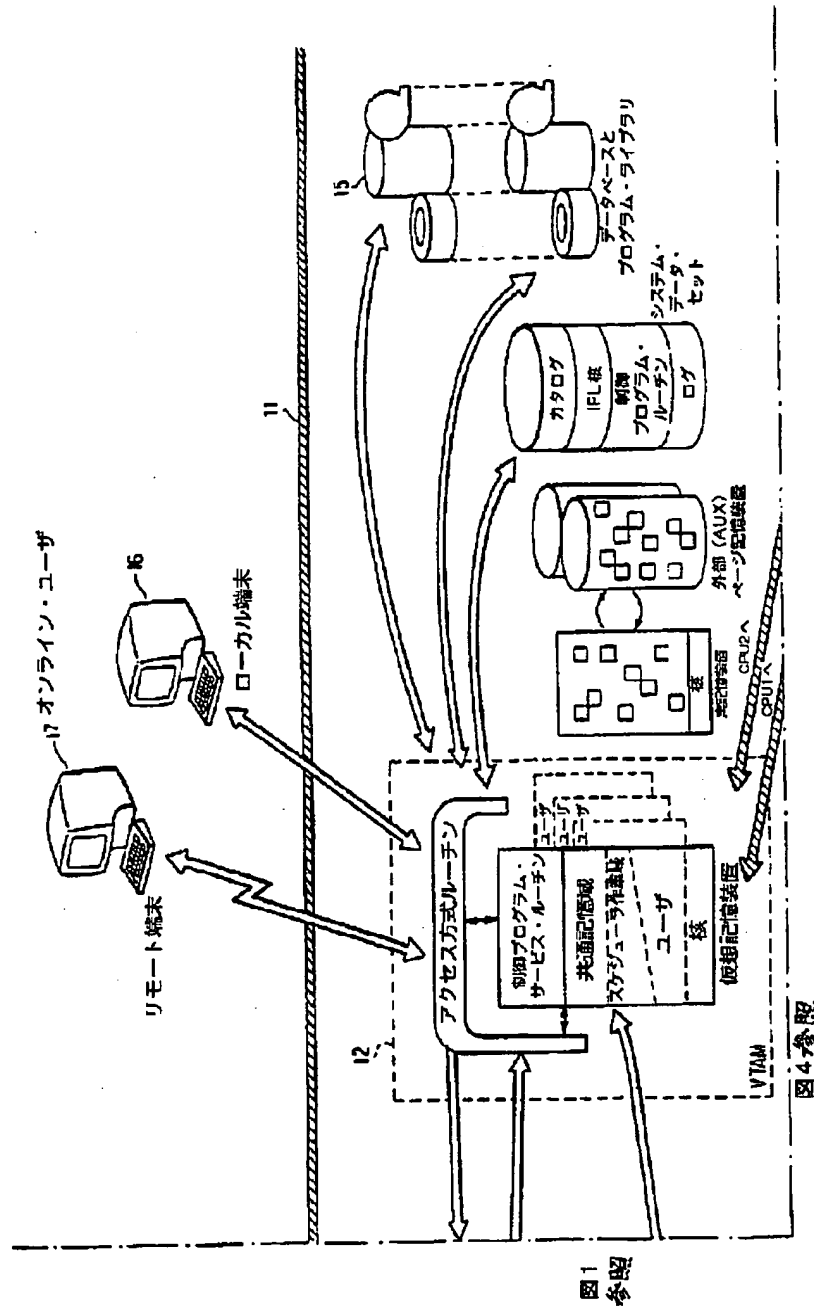
【符号の説明】

- 11 メインフレーム・コンピュータ
- 12 仮想端末アクセス方式 (VTAM)
- 13 ジョブ入力システム (TES)
- 15 直接アクセス記憶装置 (DASD)
- 16 ローカル端末
- 17 リモート端末
- 18 ローカル・ジョブ入力/出力 (I/O) 装置
- 19 リモート・ジョブI/O装置
- 21 マスタ・スケジューラ
- 22 中央コンソール
- 23 機能コンソール
- 24 ジョブ・スケジューラ
- 25 タスク・スーパーバイザ
- 26 プログラム・マネージャ
- 27 タイマ・スーパーバイザ
- 28 仮想記憶マネージャ
- 29 実記憶マネージャ
- 30 補助 (またはページ) 記憶マネージャ
- 31 復元終了マネージャ
- 34 割込みハンドラ
- 35 システム・リソース・マネージャ
- 36 ディスパッチャ
- 51 第1コンピュータ
- 52 第2コンピュータ
- 53 通信装置
- 54 オペレーティング・システム (OS)
- 55 ユーザAのアドレス空間
- 56 P₁ のアドレス空間
- 57 CMGR_Aのアドレス空間
- 62 ユーザBのアドレス空間
- 63 P₂ のアドレス空間

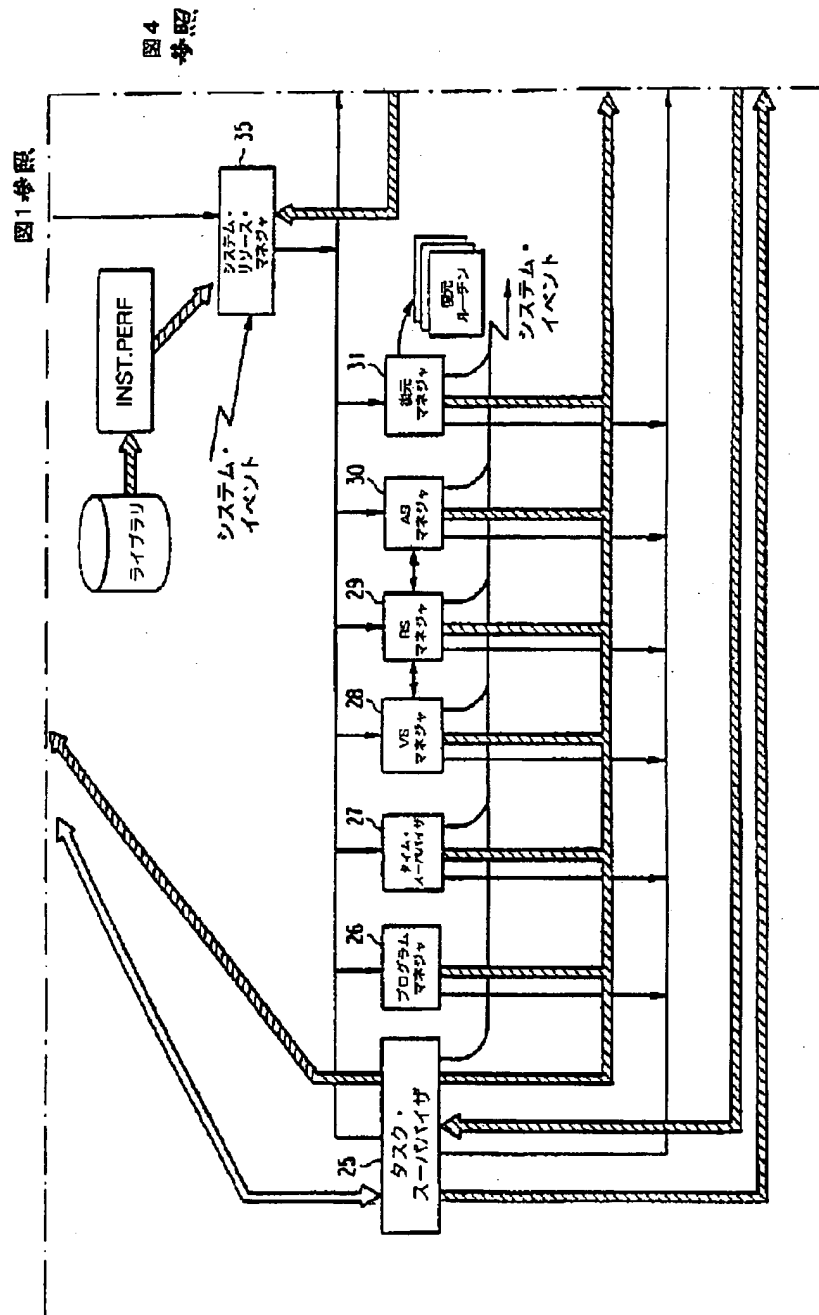
【図1】

図2
参照図3
参照

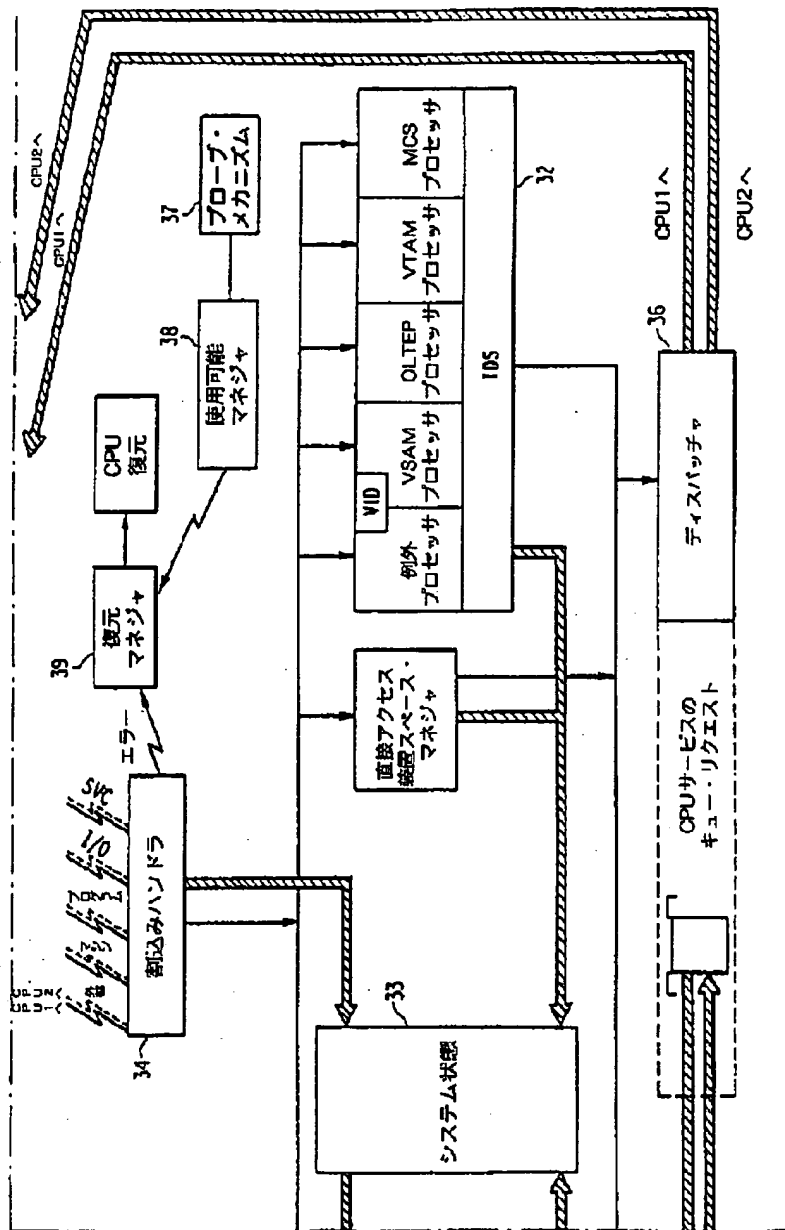
【図2】



【図3】



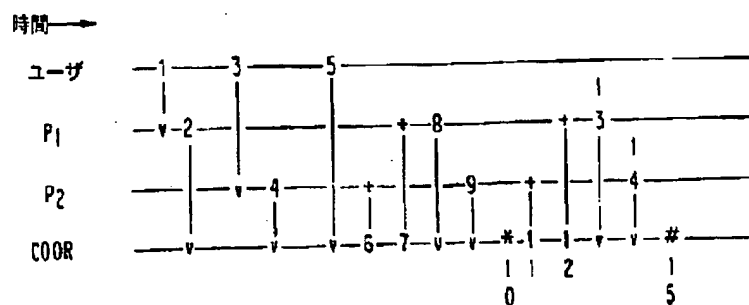
【图4】



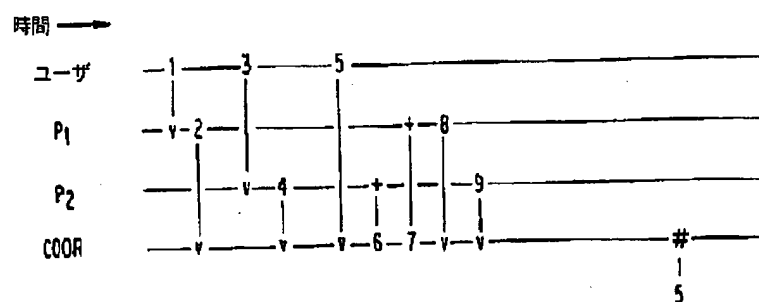
監製 張國良

圖 3

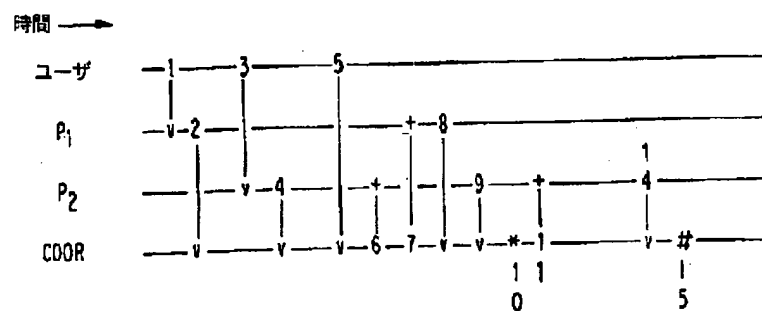
【図5】



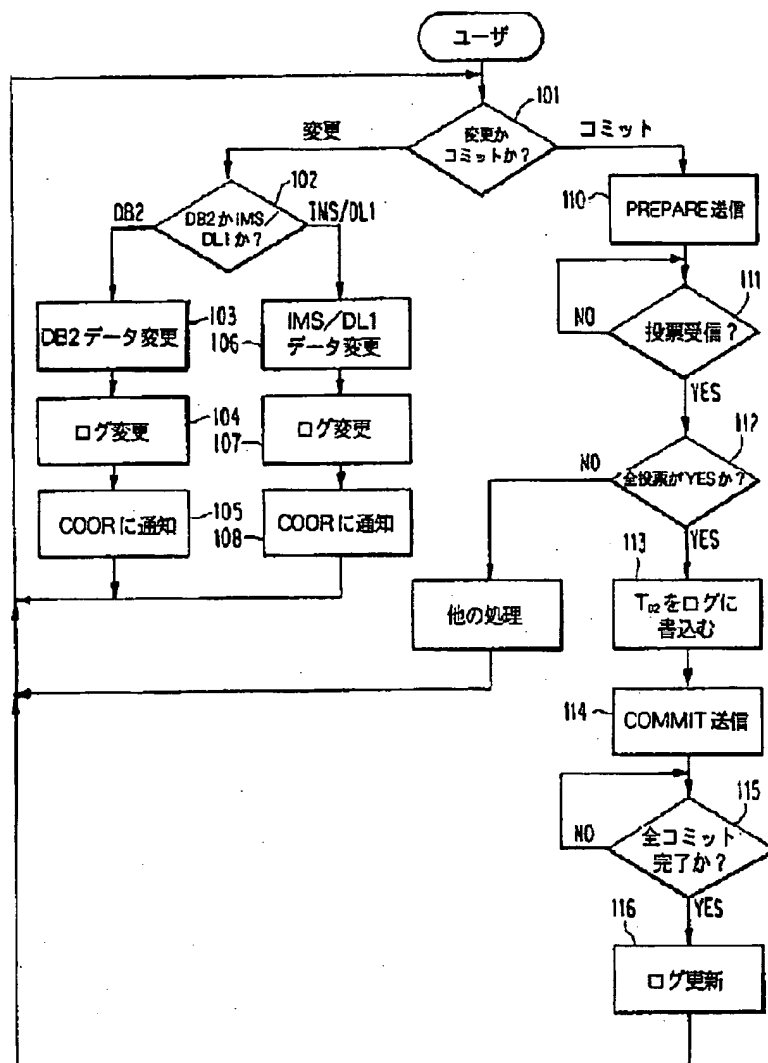
【図6】



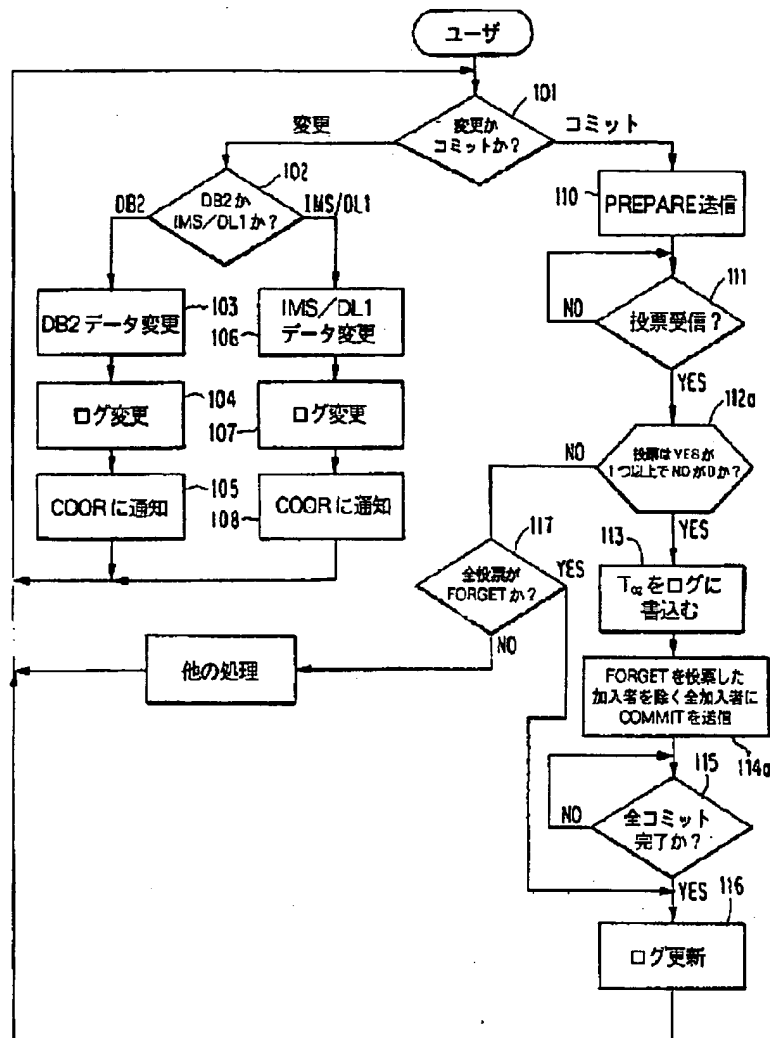
【図7】



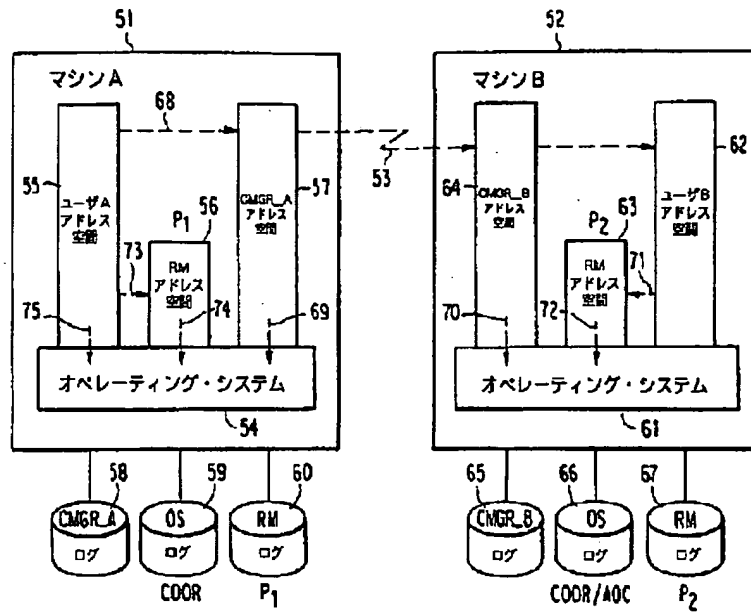
【図8】



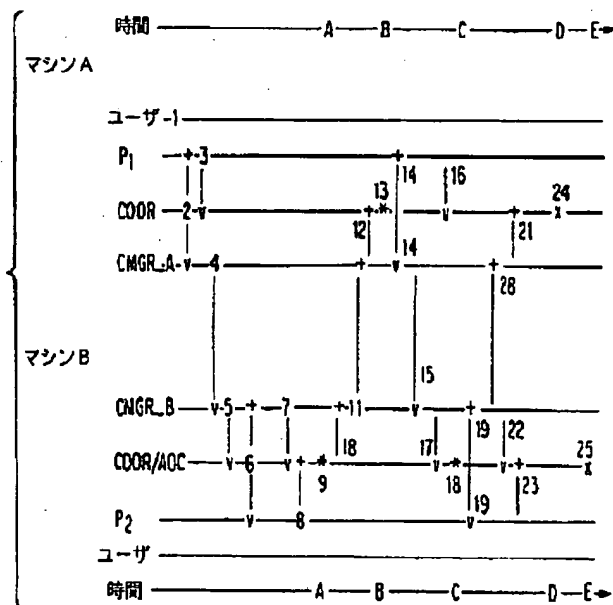
【図9】



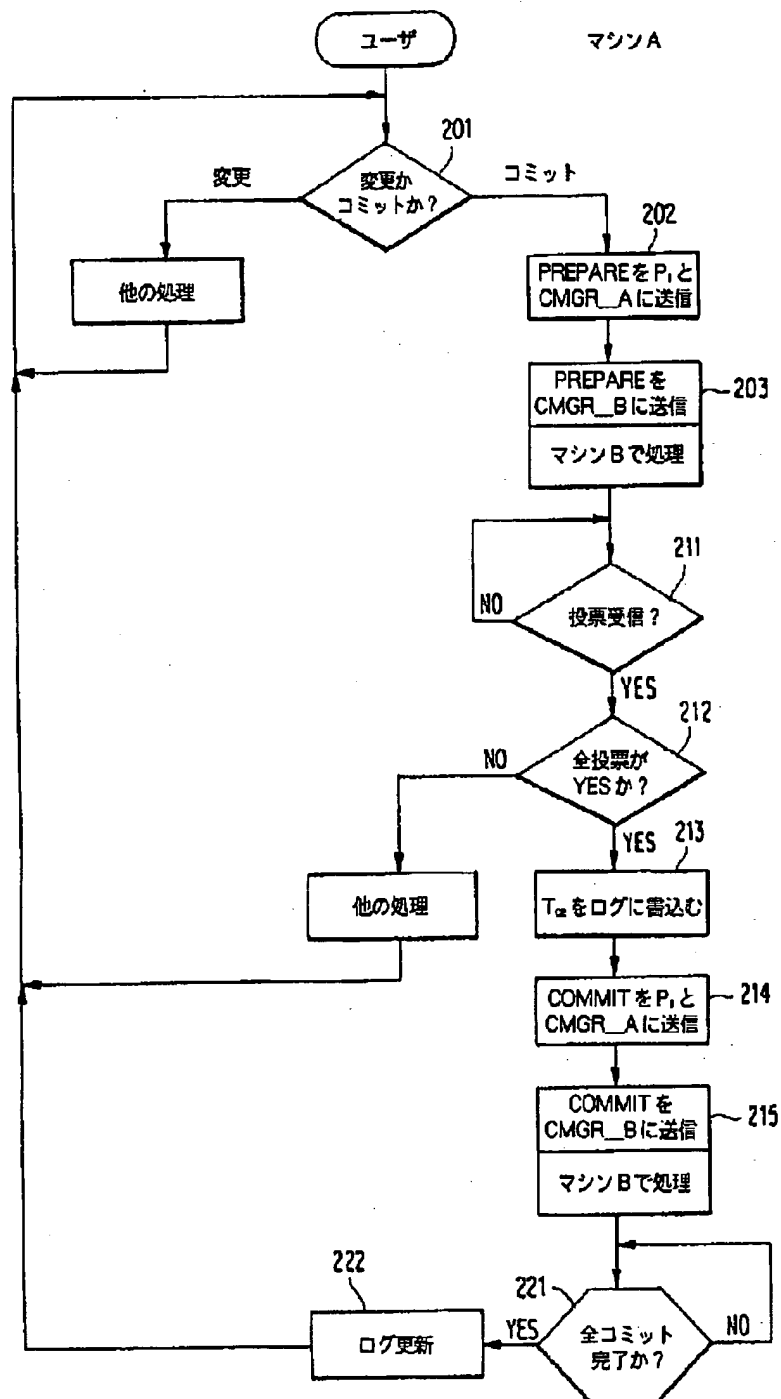
【図10】



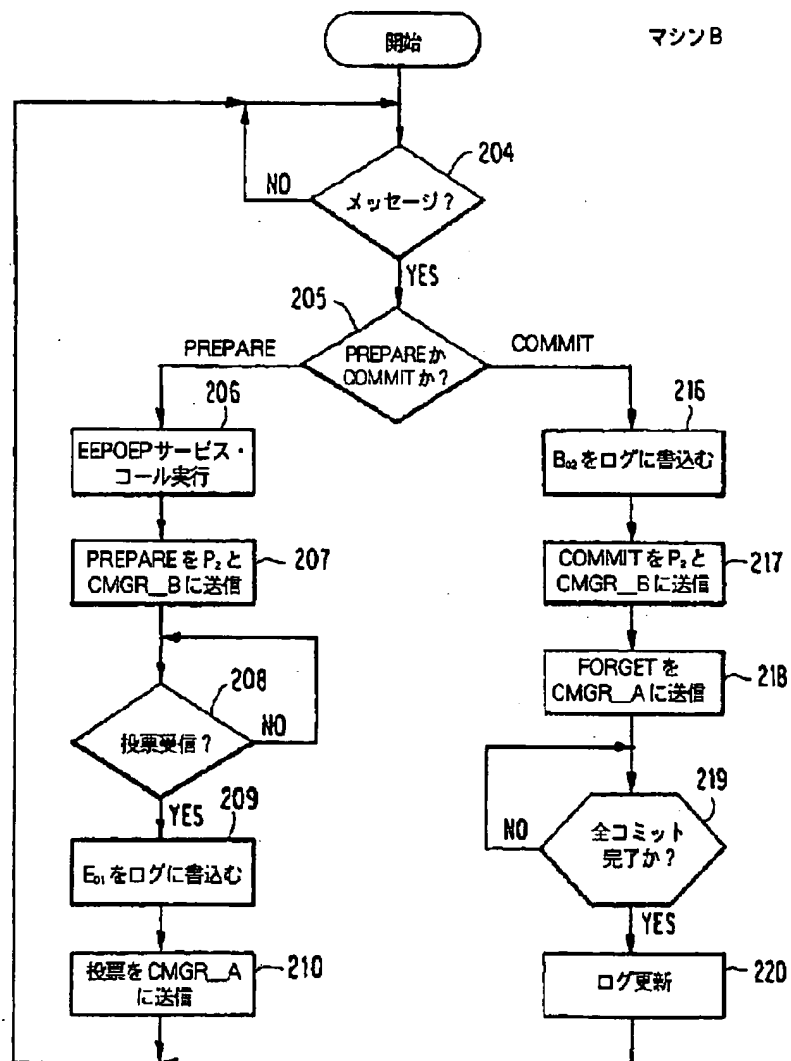
【図11】



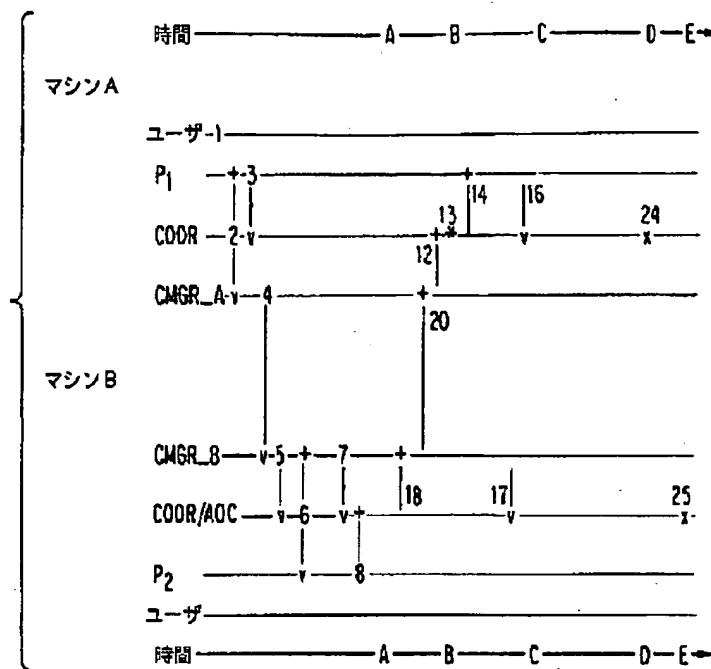
【図12】



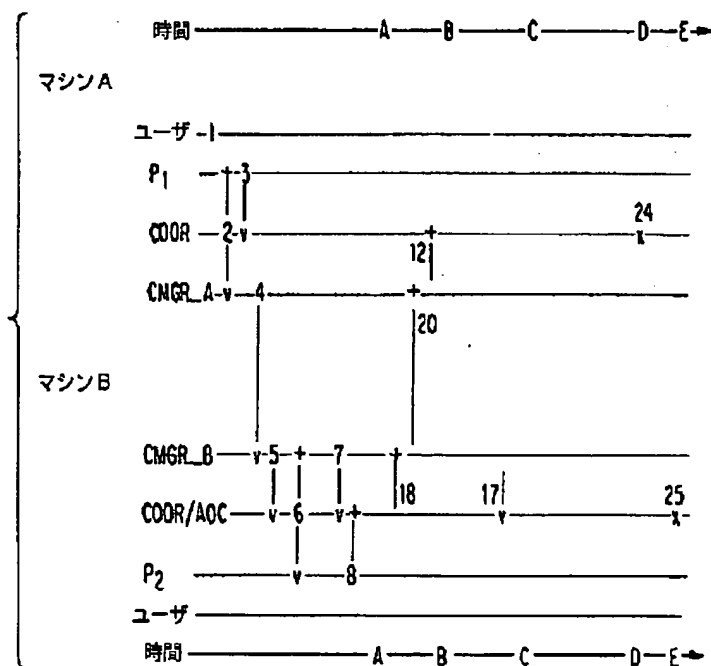
【図13】



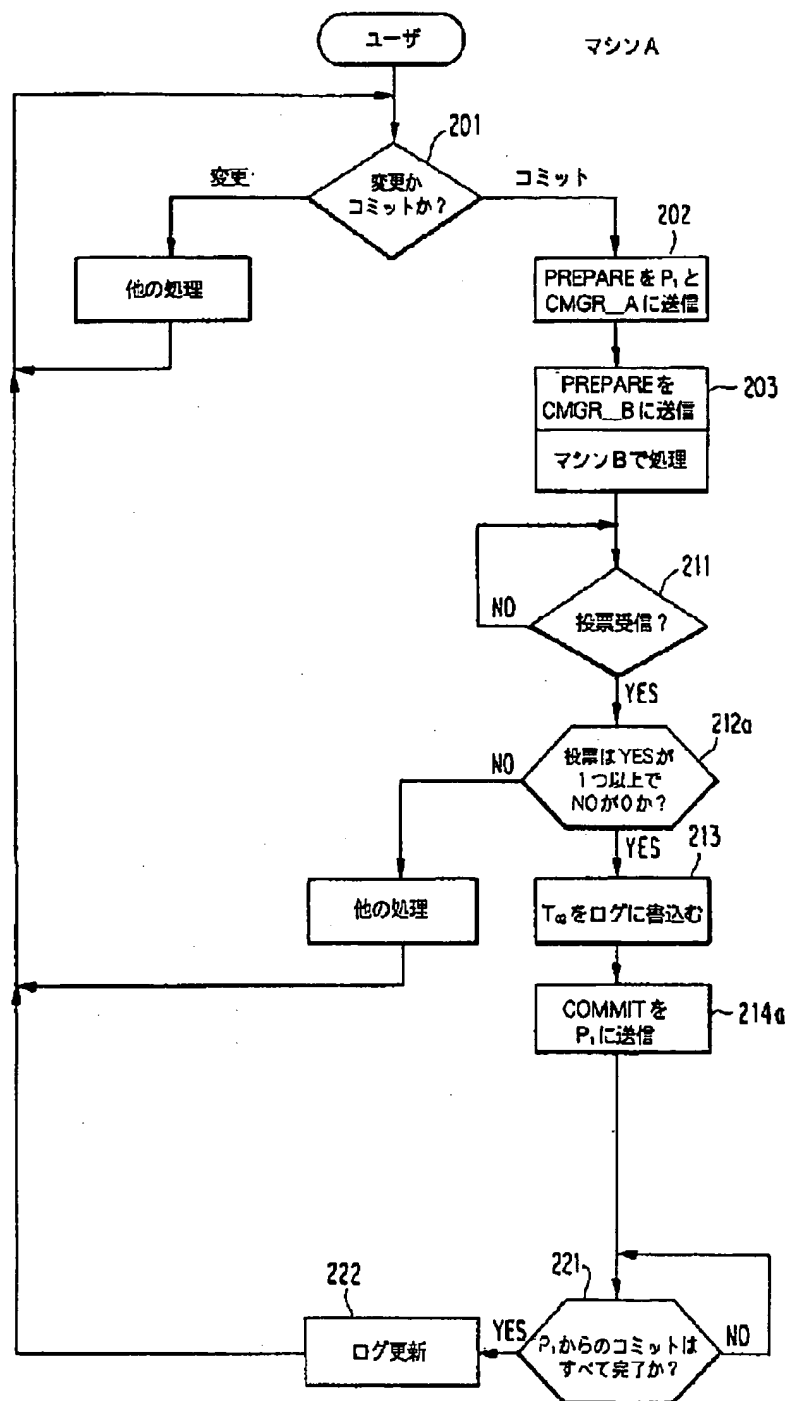
【図14】



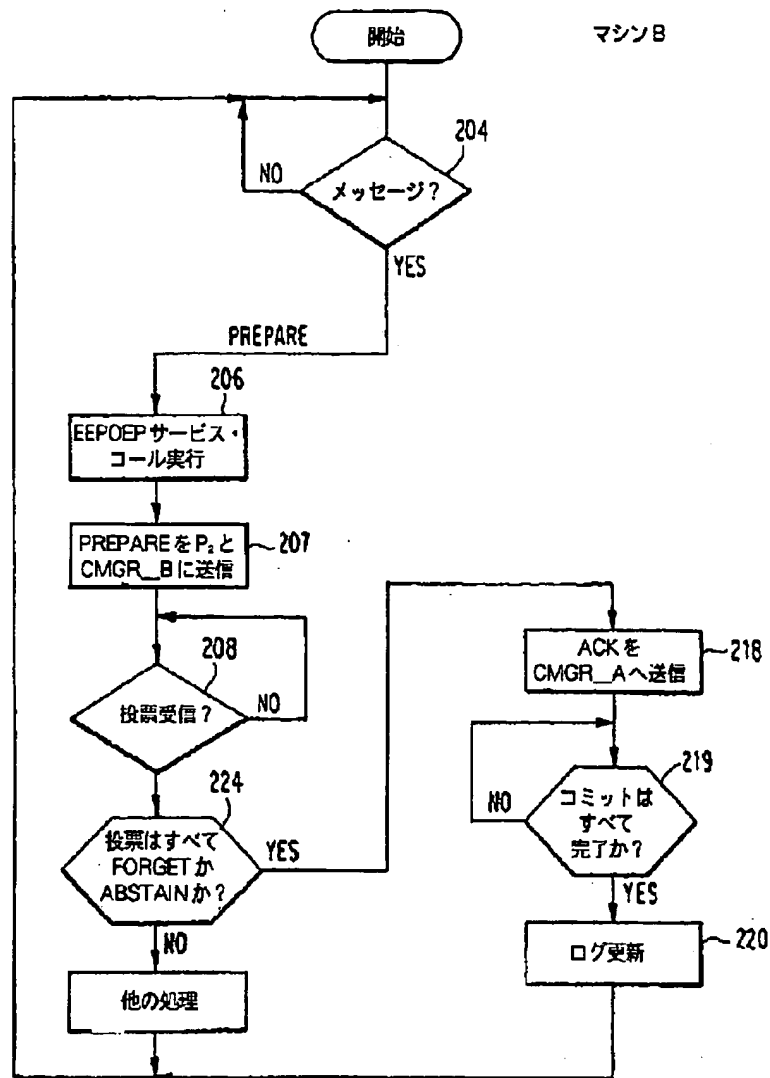
【図15】



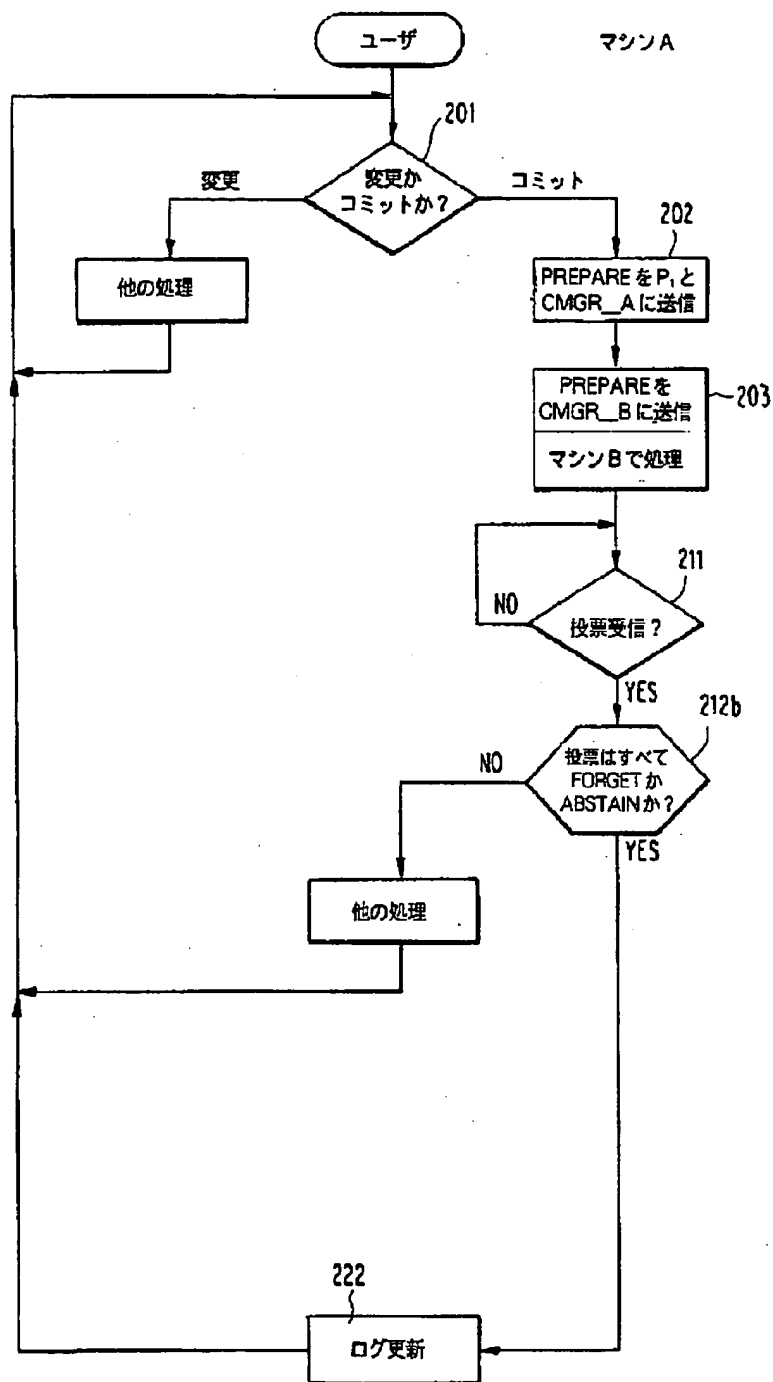
【図16】



【図17】



【図18】



フロントページの続き

(72)発明者 リチャード・ダイベンドルフ
アメリカ合衆国94040、カリフォルニア州
マウンテン・ビュー、チャタム・ウェイ
345

(72)発明者 ダニエル・エドワード・ハウス
カナダ国、オンタリオ州スカボロ、ナンバ
ー1907、グレイストーン・ウォーク・ドラ
イブ 5

(72)発明者 アール・エイチ・ジェナー
アメリカ合衆国95135、カリフォルニア州
サン・ホセ、ガロウェイ・ドライブ 7688
(72)発明者 マーガレット・ケリー・ラベル
アメリカ合衆国12603、ニューヨーク州ボ
キプシ、フッカー・アベニュー 310

(72)発明者 マイケル・ジェラルド・モール
アメリカ合衆国12540、ニューヨーク州ラ
グレーンジビル、スクエア・ウッズ・ドラ
イブ 20
(72)発明者 スチュアート・ルイス・サイレン
アメリカ合衆国95037、カリフォルニア州
モーガン・ヒル、グリフィス・ウェイ
15630